

## 明 細 書

記録媒体、再生装置、記録方法、プログラム、再生方法

### 技術分野

本発明は、動画ストリーム及びグラフィクスストリームが多重化されたデジタルストリームが記録されている記録媒体と、そのデジタルストリームを再生する再生装置とに関し、特に動画ストリーム及びグラフィクスストリームを再生装置が別個にデコードし、そのデコードにより得られた映像－グラフィクスを合成することで、再生映像を得る技術に関する。

### 背景技術

10 上述した合成技術は、再生装置における言語設定やディスプレイ設定に応じて、グラフィクスを選択して表示することもできる。また必要に応じてグラフィクスを表示したり、消したりという選択も可能である。これらの選択が可能なので、かかる技術は、DVD-Video規格や ETSI EN 300 743標準規格のプレーヤモデルに採用されている。上記  
15 規格技術においてグラフィクスストリームは、PESパケットの配列であり、PESパケットには、制御情報を格納したもの、グラフィクス本体たるグラフィクスデータを格納したものの2種類がある。制御情報は、グラフィクスデータより前に配置されており、制御情報及びグラフィクスデータの一对で一個のグラフィクス表示を実現する。具体的  
20 にいうと、再生装置は制御情報及びグラフィクスデータが順次読み込み、制御情報を解読すると共に、グラフィクスデータをデコードし、デコードにより得られた非圧縮グラフィクスを、制御情報の解読結果に従い、所望の表示タイミングで表示する。

以上のようなグラフィクス表示にあたっては、グラフィクス表示の  
25 度に、グラフィクスデータのデコードを行う必要があるので、1つのグラフィクス表示から、次のグラフィクス表示までの間隔は、グラフィクスデータのデコード期間に依存したものとなる。ここでグラフィクスの解像度が高く、デコード期間が長ければ長いほど、グラフィクスの更新間隔も長くなってしまう。

30 映画における字幕のように、グラフィクスが2～3秒という時間間隔

で更新されるのであれば、多少、グラフィックスの解像度が高く、デコード期間が長くなったとしても問題はない。しかし、アミューズメントの用途にグラフィックスを使用しようとする場合、グラフィックスの表示間隔をもっと短くしたいという要望がでてくる。

- 5      ここでアミューズメントの用途とは、登場人物の台詞を示すグラフィックスに、独特の動きを与えて、視聴者の注意をひいたり、色彩を派手に変化させることで、視聴者の意表を突くというものである。かかるグラフィックス表示の実現は、バラエティ色が濃い映像コンテンツの制作現場からの要望が特に強い。
- 10      しかしグラフィックスの滑らかな動きを実現するには、グラフィックスをデコードし、表示するという処理を、ディスプレイの一表示期間(NTSC方式の場合は1/29.97sec)という僅かな期間が経過する毎に、行う必要がある。ディスプレイの一表示期間が経過する度に、グラフィックスのデコードー表示を繰り返すというのは、多大な負荷であり、低
- 15      コストでの普及を想定しているような再生装置のハードウェアスペックでは実現が困難である。

- 無論、上述したようなグラフィックス動きや色彩変化は、動画像の一コマ一コマにグラフィックスを予め合成しておき、各コマの絵柄の一部としてグラフィックスを取り扱うことで実現可能である。しかし一コマ
- 20      一コマに、グラフィックスを予め合成しておくといく方法は、再生装置における言語設定やディスプレイ設定に応じて、グラフィックスを選択的に表示するという、融通性を欠く。そのため、かかる方法を採用している限り、将来への展望は見えてこない。

#### 発明の開示

- 25      本発明の目的は、グラフィックスを、動画のような滑らかさで動かすことができる記録媒体を提供することである。

- 上記目的は、動画ストリームとグラフィックスストリームとを多重化することにより得られたデジタルストリームが記録されており、グラフィックスストリームは、パケットの配列であり、パケットには、グラフィックスデータを格納したものと、制御情報を格納したものとがあり、
- 30

制御情報は、パケット列において自身より前方に存在するグラフィクスデータを、所定のタイミングで動画ストリームと合成して表示する旨を示す、ことを特徴とする記録媒体により達成される。

5 制御情報は、自身より前方に存在するグラフィクスデータを用いた表示を行う旨を示している。そのため、新たな座標を示す制御情報のみを再生装置に送り込むだけでグラフィクスの表示位置を変えたり、グラフィクスの色彩を変化させるという制御を再生装置に行わせることができる。

10 制御情報を送り込むだけで、グラフィクス表示を更新することができるので、グラフィクスの頭出し位置と、動画との同期が容易になる。

ここでグラフィクスの表示位置を変化させる場合、グラフィクスそのものの絵柄の変化は必要でないことが多い。人間の目は、動く対象をそれ程鮮明に捉えることができないからである。同じグラフィクスを、高速に位置を変えながら表示するという処理に適した技術が、本  
15 発明の本質である。

#### 図面の簡単な説明

図1は、本発明に係る記録媒体の、使用行為についての形態を示す図である。

図2は、BD-ROMの構成を示す図である。

20 図3は、AVClipがどのように構成されているかを模式的に示す図である。

図4(a)は、プレゼンテーショングラフィクスストリームの構成を示す図である。

25 図4(b)は、機能セグメントを変換することで得られるPESパケットを示す図である。

図5は、様々な種別の機能セグメントにて構成される論理構造を示す図である。

図6は、字幕の表示位置と、Epochとの関係を示す図である。

30 図7(a)は、ODSによるグラフィクスオブジェクトの定義を示す図である。

図 7 (b) は、PDSのデータ構造を示す図である。

図 8 (a) は、WDSのデータ構造を示す図である。

図 8 (b) は、PCSのデータ構造で構成される。

図 9 は、字幕表示を実現するためのDisplay Setの記述例である。

5 図 10 は、DS1におけるWDS、PCSの記述例を示す図である。

図 11 は、DS2におけるPCSの記述例を示す図である。

図 12 は、DS3におけるPCSの記述例を示す図である。

図 13 は、図 10～図 12 に示すようなグラフィクスアップデート  
を実現するにあたっての、オブジェクトバッファにおけるメモリ空間  
10 を示す図である。

図 14 は、decode\_durationの計算アルゴリズムの一例を示す図で  
ある。

図 15 は、図 14 のプログラムのアルゴリズムを図式化したフロー  
チャートである。

15 図 16 (a) (b) は、図 14 のプログラムのアルゴリズムを図式  
化したフローチャートである。

図 17 (a) は、1つのwindowに1つのODSが存在するケースを想定  
した図である。

図 17 (b) (c) は、図 14 で引用した各数値の時間的な前後関  
20 係を示すタイミングチャートである。

図 18 (a) は、1つのwindowに2つのODSが存在するケースを想定  
した図である。

図 18 (b) (c) は、図 14 で引用した各数値の時間的な前後関  
係を示すタイミングチャートである。

25 図 19 (a) は、2つのwindowのそれぞれに、ODSが1つずつ存在す  
るケースを想定したタイミングチャートである。

図 19 (b) は、デコード期間(2)がクリア期間(1)+書込期間(31)  
より長くなるケースを示すタイミングチャートである。

図 19 (c) は、クリア期間(1)+書込期間(31)がデコード期間(2)  
30 より長くなるケースを示すタイミングチャートである。

図 2 0 は、前方参照を行う PCS を描いた図である。

図 2 1 (a) は、グラフィックスが画面内を動き回るという表示効果を実現する Epoch を示す図である。

図 2 1 (b) は、DS1~DS8 に含まれる PCS の内容と、PTS 値とを示す図である。 図 2 2 (a) は、DS0 内の ODS を示す図である。

図 2 2 (b) は、座標 (x1, y1) (x2, y2) (x3, y3) ... (x8, y8) が、ウィンドウの座標系のどこに存在するかを示す図である。

図 2 3 は、各 Display Set に属する各機能セグメントのタイムスタンプに対する設定を示す図である。

10 図 2 4 は、アップデートの、時間的変遷の具体例を示す図である。

図 2 5 は、Pallet Only Update を実現する一連の Display Set を示す図である。

図 2 6 (a) は、DS0 内の PDS 及び各 Display Set の PCS がどのような内容であるかを示す図である。

15 図 2 6 (b) は、DS0~DS3 内の PCS を示す図である。

図 2 7 は、4 つの Display Set が読み込ませることで実現される、表示効果を現した図である。

図 2 8 は、本発明に係る再生装置の内部構成を示す図である。

図 2 9 は、書込レート Rx, Rc, Rd、グラフィックスプレーン 8、Coded  
20 Data Buffer 1 3、Object Buffer 1 5、Composition Buffer 1 6 のサイズを示す図である。

図 3 0 は、再生装置によるパイプライン処理を示すタイミングチャートである。

図 3 1 は、ODS のデコードが、グラフィックスプレーンのクリアより早く終わる場合を想定したパイプライン処理を示すタイミングチャートである。

図 3 2 は、Composition バッファ 1 6、Object Buffer 1 5、Coded Data バッファ 1 3、グラフィックスプレーン 8 における蓄積量の時間的遷移を示すタイミングチャートである。

30 図 3 3 は、機能セグメントのロード処理の処理手順を示すフローチ

ャートである。

図 3 4 は、多重化の一例を示す図である。

図 3 5 は、DS10が再生装置のCoded Data Buffer 1 3 にロードされる様子を示す図である。

5 図 3 6 は、通常再生が行われる場合を示す図である。

図 3 7 は、図 3 6 のように通常再生が行われた場合のDS1, 10, 20のロードを示す図である。

図 3 8 は、Graphical Controller 1 7 の処理手順を示すフローチャートである。

10 図 3 9 は、Graphical Controller 1 7 の処理手順を示すフローチャートである。

図 4 0 は、Graphical Controller 1 7 の処理手順を示すフローチャートである。

15 図 4 1 は、第 1 実施形態に示したPCSが記録されたBD-ROMを製造するための製造工程を示す図である。

#### 発明を実施するための最良の形態

##### (第 1 実施形態)

以降、本発明に係る記録媒体の実施形態について説明する。先ず始めに、本発明に係る記録媒体の実施行為のうち、使用行為についての形態を説明する。図 1 は、本発明に係る記録媒体の、使用行為についての形態を示す図である。図 1 において、本発明に係る記録媒体は、BD-ROM 1 0 0 である。このBD-ROM 1 0 0 は、再生装置 2 0 0、テレビ 3 0 0、リモコン 4 0 0 により形成されるホームシアターシステムに、  
25 映画作品を供給するという用途に供される。

以上が本発明に係る記録媒体の使用形態についての説明である。

続いて本発明に係る記録媒体の実施行為のうち、生産行為についての形態について説明する。本発明に係る記録媒体は、BD-ROMの応用層に対する改良により実施することができる。図 2 は、BD-ROMの構成を示す図である。  
30

本図の第4段目にBD-ROMを示し、第3段目にBD-ROM上のトラックを示す。本図のトラックは、BD-ROMの内周から外周にかけて螺旋状に形成されているトラックを、横方向に引き伸ばして描画している。このトラックは、リードイン領域と、ボリューム領域と、リードアウト領域とからなる。本図のボリューム領域は、物理層、ファイルシステム層、応用層というレイヤモデルをもつ。ディレクトリ構造を用いてBD-ROMの応用層フォーマット(アプリケーションフォーマット)を表現すると、図中の第1段目のようになる。本図に示すようにBD-ROMには、ROOTディレクトリの下にBDMVディレクトリがあり、BDMVディレクトリの配下には、AVClipを格納したファイル(XXX.M2TS)、AVClipの管理情報を格納したファイル(XXX.CLPI)、AVClipにおける論理的な再生経路(PL)を定義したファイル(YYY.MPLS)が存在する。本図に示すようなアプリケーションフォーマットを作成することにより、本発明に係る記録媒体は生産される。尚、XXX.M2TS、XXX.CLPI、YYY.MPLSといったファイルが、それぞれ複数存在する場合は、BDMVディレクトリの配下に、STREAMディレクトリ、CLIPINFディレクトリ、PLAYLISTディレクトリという3つのディレクトリを設け、STREAMディレクトリにXXX.M2TSと同じ種別のファイルを、CLIPINFディレクトリにXXX.CLPIと同じ種別のファイルを、PLAYLISTディレクトリにYYY.MPLSと同じ種別のファイルを格納することが望ましい。

このアプリケーションフォーマットにおけるAVClip(XXX.M2TS)について説明する。

AVClip(XXX.M2TS)は、MPEG-TS(Transport Stream)形式のデジタルストリームであり、ビデオストリーム、1つ以上のオーディオストリーム、プレゼンテーショングラフィクスストリームを多重化することで得られる。ビデオストリームは映画の動画部分を、オーディオストリームは映画の音声部分を、プレゼンテーショングラフィクスストリームは、映画の字幕をそれぞれ示している。図3は、AVClipがどのように構成されているかを模式的に示す図である。

AVClipは(中段)、複数のビデオフレーム(ピクチャp1,2,3)からな

るビデオストリーム、複数のオーディオフレームからなるオーディオストリームを(上1段目)、PESパケット列に変換し(上2段目)、更にTSパケットに変換し(上3段目)、同じくプレゼンテーショングラフィックスストリーム(下1段目)を、PESパケット列に変換し(下2段目)、  
5 更にTSパケットに変換して(下3段目)、これらを多重化することで構成される。

本図において多重されるプレゼンテーショングラフィックスストリームは1つであるが、BD-ROMが多言語対応型である場合、その言語毎のプレゼンテーショングラフィックスストリームがAVClipに多重され  
10 る。かかる過程を経て生成されたAVClipは、通常のコンピュータファイル同様、複数のエクステンツに分割され、BD-ROM上の領域に記録される。

続いてプレゼンテーショングラフィックスストリームについて説明する。図4(a)は、プレゼンテーショングラフィックスストリームの構成を示す図である。第1段目は、AVClipを構成するTSパケット列を示す。第2段目は、グラフィックスストリームを構成するPESパケット列を示す。第2段目におけるPESパケット列は、第1段目におけるTSパケットのうち、所定のPIDをもつTSパケットからペイロードを取り出して、連結することにより構成される。

20 第3段目は、グラフィックスストリームの構成を示す。グラフィックスストリームは、PCS(Presentation Composition Segment)、WDS(Window Define Segment)、PDS(Palette Definition Segment)、ODS(Object Definition Segment)、END(END of Display Set Segment)と呼ばれる機能セグメントからなる。これらの機能セグメントのうち、  
25 PCSは、画面構成セグメントと呼ばれ、WDS、PDS、ODS、ENDは定義セグメントと呼ばれる。PESパケットと機能セグメントとの対応関係は、1対1の関係、1対多の関係である。つまり機能セグメントは、1つのPESパケットに変換されてBD-ROMに記録されるか、又は、フラグメント化され、複数PESパケットに変換されてBD-ROMに記録される。

30 図4(b)は、機能セグメントを変換することで得られるPESパケ



ットを示す図である。図4(b)に示すようにPESパケットは、パケットヘッダと、ペイロードとからなり、このペイロードが機能セグメント実体にあたる。またパケットヘッダには、この機能セグメントに対応するDTS、PTSが存在する。尚以降の説明では、機能セグメントが格納されるPESパケットのヘッダ内に存在するDTS及びPTSを、機能セグメントのDTS及びPTSとして扱う。

これら様々な種別の機能セグメントは、図5のような論理構造を構築する。図5は、様々な種別の機能セグメントにて構成される論理構造を示す図である。本図は第3段目に機能セグメントを、第2段目にDisplay Setを、第1段目にEpochをそれぞれ示す。

第2段目のDisplay Set(DSと略す)とは、グラフィクスストリームを構成する複数機能セグメントのうち、一画面分のグラフィクスを構成するものの集合をいう。図中の破線は、第3段目の機能セグメントが、どのDSに帰属しているかという帰属関係を示す。

PCS-WDS-PDS-ODS-ENDという一連の機能セグメントが、1つのDSを構成していることがわかる。再生装置は、このDSを構成する複数機能セグメントをBD-ROMから読み出せば、一画面分のグラフィクスを構成することができる。

第1段目のEpochとは、AVClipの再生時間軸上においてメモリ管理の連続性をもっている一つの期間、及び、この期間に割り当てられたデータ群をいう。ここで想定しているメモリとは、一画面分のグラフィクスを格納しておくためのグラフィクスプレーン、伸長された状態のグラフィクスデータを格納しておくためのオブジェクトバッファである。これらについてのメモリ管理に、連続性があるというのは、このEpochにあたる期間を通じてこれらグラフィクスプレーン及びオブジェクトバッファのフラッシュは発生せず、グラフィックスプレーン内のある決められた矩形領域内でのみ、グラフィクスの消去及び再描画が行われることをいう(※ここでフラッシュとは、プレーン及びバッファの格納内容を全部クリアしてしまうことである。)。この矩形領域の縦横の大きさ及び位置は、Epochにあたる期間において、終

始固定されている。グラフィックスプレーンにおいて、この固定化された領域内で、グラフィックスの消去及び再描画を行っている限り、映像とグラフィックスとの同期が保障される。つまりEpochは、映像ーグラフィックスの同期の保障が可能な再生時間軸上の一単位ということ

5 ができる。グラフィックスプレーンにおいて、グラフィックスの消去・再描画を行うべき領域を変更したい場合は、再生時間軸上においてその変更時点を定義し、その変更時点以降を、新たなEpochにせねばならない。この場合、2つのEpochの境界では、映像ーグラフィックスの同期は保証されない。

10 Epochにおける字幕の位置関係にたとえば、再生時間軸上において、画面上のある決まった矩形領域内に字幕が出現している期間が、Epochということができる。図6は、字幕の表示位置と、Epochとの関係を示す図である。本図では、動画の各ピクチャの絵柄に応じて字幕の位置を変更するという配慮がなされている。つまり5つの字幕「本当

15 は」「ウソだった」「ごめん」「あれから」「3年がたった」のうち、3つの字幕「本当は」「ウソだった」「ごめん」は画面の下側に、「あれから」「3年がたった」は画面の上側に配置されている。これは画面の見易さを考え、画面中の余白にあたる位置に字幕を配置することを意図している。かかる時間的な変動がある場合、AVClipの再生時間軸において、下側の

20 余白に字幕が出現している期間が1つのEpoch1、上側の余白に字幕が出現している期間が別のEpoch2になる。これら2つのEpochは、それぞれ独自の字幕の描画領域をもつことになる。Epoch1では、画面の下側の余白が字幕の描画領域(window1)になる。一方Epoch2では、画面の上側の余白が字幕の描画領域(window2)になる。これらのEpoch1,2に

25 において、バッファ・プレーンにおけるメモリ管理の連続性は保証されているので、上述した余白での字幕表示はシームレスに行われる。以上がEpochについての説明である。続いてDisplay Setについて説明する。

図5における破線hk1,2は、第2段目の機能セグメントが、どの

30 Epochに帰属しているかという帰属関係を示す。Epoch

Start, Acquisition Point, Normal Caseという一連のDSは、第1段目のEpochを構成していることがわかる。『Epoch Start』、『Acquisition Point』、『Normal Case』は、DSの種類である。本図におけるAcquisition Point、Normal Caseの順序は、一例にすぎず、どちらが先であってもよい。

『Epoch Start』は、“新表示”という表示効果をもたらすDSであり、新たなEpochの開始を示す。そのためEpoch Startは、次の画面合成に必要な全ての機能セグメントを含んでいる。Epoch Startは、映画作品におけるチャプター等、頭出しがなされることが判明している位置に配置される。

『Acquisition Point』は、“表示リフレッシュ”という表示効果をもたらすDisplay Setであり、先行するEpoch Startと全く同じDisplay Setをいう。Acquisition PointたるDSは、Epochの開始時点ではないが、次の画面合成に必要な全ての機能セグメントを含んでいるので、Acquisition PointたるDSから頭出しを行えば、グラフィックス表示を確実に実現することができる。つまりAcquisition PointたるDSは、Epochの途中からの画面構成を可能するという役割をもつ。

Acquisition PointたるDisplay Setは、頭出し先になり得る位置に組み込まれる。そのような位置には、タイムサーチにより指定され得る位置がある。タイムサーチとは、何分何秒という時間入力をユーザから受け付けて、その時間入力に相当する再生時点から頭出しを行う操作である。かかる時間入力は、10分単位、10秒単位というように、大まかな単位でなされるので、10分間隔の再生位置、10秒間隔の再生位置がタイムサーチにより指定され得る位置になる。このようにタイムサーチにより指定され得る位置にAcquisition Pointを設けておくことにより、タイムサーチ時のグラフィックスストリーム再生を好適に行うことができる。

『Normal Case』は、“表示アップデート”という表示効果をもたらすDSであり、前の画面合成からの差分のみを含む。例えば、あるDSvの字幕は、先行するDSuと同じ内容であるが、画面構成が、この先行

するDSuとは異なる場合、PCSと、ENDのみのDSvを設けてこのDSvを Normal CaseのDSにする。こうすれば、重複するODSを設ける必要はなくなるので、BD-ROMにおける容量削減に寄与することができる。一方、 Normal CaseのDSは、差分にすぎないので、Normal Case単独では画面

5 構成は行えない。

続いてDefinition Segment(ODS, WDS, PDS)について説明する。

『Object\_Definition\_Segment』は、グラフィクスオブジェクトを定義する機能セグメントである。このグラフィクスオブジェクトについて以下説明する。BD-ROMに記録されているAVClipは、ハイビジョン

10 並みの高画質をセールスポイントにしているため、グラフィクスオブジェクトの解像度も、1920×1080画素という高精細な大きさに設定されている。1920×1080という解像度があるので、BD-ROMでは、劇場上映用の字幕の字体、つまり、手書きの味わい深い字体の字幕表示を鮮やかに再現できる。グラフィクスオブジェクトは複数のランレングス

15 データからなる。ランレングスデータとは、画素値を示すPixel Codeと、画素値の連続長とにより、画素列を表現したデータである。Pixel Codeは、8ビットの値であり、1～255の値をとる。ランレングスデータでは、このPixel Codeによりフルカラーの16,777,216色から任意の256色を選んで画素の色として設定することができる。尚、字幕として

20 表示される場合、グラフィクスオブジェクトは、透明色の背景に、文字列を配置することで描画せねばならない。

ODSによるグラフィクスオブジェクトの定義は、図7(a)に示すようなデータ構造をもってなされる。ODSは、図7(a)に示すように自身がODSであることを示す『segment\_type』と、ODSのデータ長を示す『segment\_length』と、EpochにおいてこのODSに対応するグラフィクスオブジェクトを一意に識別する『object\_id』と、EpochにおけるODSのバージョンを示す『object\_version\_number』と、

25

『last\_in\_sequence\_flag』と、グラフィクスオブジェクトの一部又は全部である連続バイト長データ『object\_data\_fragment』とからなる。

30

『object\_id』は、ODSはデコードされて対応するグラフィクスオブジェクトがオブジェクトバッファ上に読み出された場合、このグラフィクスオブジェクトと、オブジェクトバッファにおいてこのグラフィクスオブジェクトが占有している領域とを一意に識別する識別子で

5 ある。1つ以上のグラフィクスオブジェクトが格納された状態において、オブジェクトバッファのメモリ空間における個々の領域は、このobject\_idにより識別されることになる。仮にあるobject\_idが2つ以上のDisplay Setに付加されている場合、先行するODSに対応するグラフィクスオブジェクトは、オブジェクトバッファに格納された後、同  
10 じobject\_idを有する後続のグラフィクスオブジェクトにより上書きされることになる。かようなアップデートは、空き領域の虫食い状の発生や、グラフィクスオブジェクトの離散配置を避けるとの配慮である。その理由は以下の通りである。グラフィクス表示を行うため、オブジェクトバッファ上のグラフィクスオブジェクトは絶えずグラフィ  
15 ックスプレーンに転送されることになる。オブジェクトバッファ内に空き領域が虫食い状に発生したり、また同じobject\_idをもったグラフィクスオブジェクトがバラバラにされて離散的に配置されると、グラフィクスオブジェクトを読み出すためのオーバヘッドが発生し、オブジェクトバッファ→グラフィックスプレーンの転送時にあたっ  
20 ての転送効率が落ちる。この効率低下は、グラフィクスー動画の同期表示に悪影響を及ぼしかねない。かかる事情から、オブジェクトバッファ上においてあるobject\_idで識別されるグラフィクスオブジェクトが共有している領域は、同じ大きさと同じobject\_idとをもったグラフィクスオブジェクトでのみ上書きできるとしている。

25 グラフィクスオブジェクト上書きにあたって、後続するグラフィクスオブジェクトのサイズは、先行するものと同じサイズでなければならない。先行するグラフィクスオブジェクトにより小さすぎてもいけないし、大き過ぎてもいけない。アップデートにあたってグラフィクスオブジェクトのサイズが同じになるようにグラフィクスオブジェ  
30 クトを作成しておくというのは、オーサリング担当者にとっての責務

になる。同じIDをもつグラフィクスオブジェクトの縦横サイズを同じにすると、1つのEpoch内の制限に過ぎない。あるEpochに属するグラフィクスオブジェクトと、次のEpochに属するグラフィクスオブジェクトとでは縦横のサイズが変わっても良い。

- 5 『last\_in\_sequence\_flag』、『object\_data\_fragment』について説明する。PESパケットのペイロードの制限から、字幕を構成する非圧縮グラフィクスが1つのODSでは格納できない場合がある。そのような場合、グラフィクスを分割することにより得られた1部分(フラグメント)がobject\_data\_fragmentに設定される。1つのグラフィクスオブジェクトを複数ODSで格納する場合、最後のフラグメントを除く全てのフラグメントは同じサイズになる。つまり最後のフラグメントは、それ以前のフラグメントサイズ以下となる。これらフラグメントを格納したODSは、DSにおいて同じ順序で出現する。グラフィクスオブジェクトの最後は、last\_in\_sequence\_flagをもつODSにより指示される。
- 10 上述したODSのデータ構造は、前のPESパケットからフラグメントを詰めてゆく格納法を前提にしているが、各PESパケットに空きが生じるように、詰めてゆくという格納法であっても良い。以上がODSの説明である。

- 『Palette Definition Segment(PDS)』は、色変換用のパレットを定義する情報である。パレットとは、1~255のPixel Codeと、画素値との組合せを示すデータである。ここで画素値は、赤色差成分(Cr値)、青色差成分(Cb値)、輝度成分(Y値)、透明度(T値)から構成される。各ランレングスデータが有するPixel Codeを、パレットに示される画素値に置き換えることで、ランレングスデータは発色されることになる。
- 20 PDSのデータ構造を図7(b)に示す。図7(b)に示すようにPDSは、自身がPDSであることを示す『segment\_type』、PDSのデータ長を示す『segment\_length』、このPDSに含まれるパレットを一意に識別する『pallet\_id』、EpochにおけるEpochのPDSのバージョンを示す『pallet\_version\_number』、各エントリーについての情報
- 25 『pallet\_entry』からなる。『pallet\_entry』は、各エントリーにお

ける赤色差成分(Cr値),青色差成分(Cb値),輝度成分Y値,透明度(T値)を示す。

続いてWDSについて説明する。

『window\_definition\_segment』は、グラフィックスプレーンの矩形領域を定義するための機能セグメントである。Epochでは、クリア及び再描画が、グラフィックスプレーンにおけるある矩形領域内で行われている場合のみ、メモリ管理に連続性が生ずることは既に述べている。このグラフィックスプレーンにおける矩形領域は"window"と呼ばれ、このWDSで定義される。図8(a)は、WDSのデータ構造を示す図である。本図に示すようにWDSは、グラフィックスプレーンにおいてウィンドウを一意に識別する『window\_id』と、グラフィックスプレーンにおける左上画素の水平位置を示す

『window\_horizontal\_position』と、グラフィックスプレーンにおける左上画素の垂直位置を示す『window\_vertical\_position』と、グラフィックスプレーンにおけるウィンドウの横幅を示す

『window\_width』と、グラフィックスプレーンにおける縦幅を示す『window\_height』とを用いて表現される。

window\_horizontal\_position、window\_vertical\_position、window\_width、window\_heightがとりうる値について説明する。これらが想定している座標系は、グラフィックスプレーンの内部領域であり、このグラフィックスプレーンは、縦:video\_height、横:video\_widthという二次元状の大きさをもつ。

window\_horizontal\_positionは、グラフィックスプレーンにおける左上画素の水平アドレスであるので、1~video\_widthの値をとり、window\_vertical\_positionは、グラフィックスプレーンにおける左上画素の垂直アドレスであるので1~video\_heightの値をとる。

window\_widthは、グラフィックスプレーンにおけるウィンドウの横幅であるので、1~video\_width-window\_horizontal\_positionの値をとり、window\_heightは、グラフィックスプレーンにおける縦幅であるので1~video\_height-window\_vertical\_positionの値をとる。

WDSのwindow\_horizontal\_position、window\_vertical\_position、window\_width、window\_heightにより、グラフィックスプレーンの何処にウィンドウを配置するか、ウィンドウの大きさをどれだけにするかをEpoch毎に規定することができる。そのため、あるEpochに属する  
5 ピクチャが表示されている期間において、ピクチャ内の絵柄の邪魔にならないように、ピクチャ上の余白にあたる位置に、ウィンドウが現れるようオーサリング時に調整しておくことができる。これによりグラフィックスによる字幕表示を見易くすることができる。WDSはEpoch毎に定義可能なので、ピクチャの絵柄に時間的な変動があっても、その  
10 変動に応じて、グラフィックスを見易く表示することができる。そのため、結果として、字幕を映像本体に組み込むのと同じレベルにまで映画作品の品質を高めることができる。

続いて『END of Display Set Segment』について説明する。END of Display Set Segmentは、Display Setの伝送の終わりを示す指標であり、Display Setにおける機能セグメントのうち、最後のODSの直後に  
15 配置される。この END of Display SetSegmentの内部構成は、自身がEND of Display SetSegmentであることを示す『segment\_type』と、当該機能セグメントのデータ長を示す『segment\_length』とからなり、これといて説明が必要な構成要素はない。故に図示は省略する。

20 以上がODS、PDS、WDS、ENDについての説明である。続いてPCSについて説明する。

PCSは、対話的な画面を構成する機能セグメントである。PCSは、図8（b）に示すデータ構造で構成される。本図に示すようにPCSは、『segment\_type』と、『segment\_length』と、『composition\_number』  
25 と、『composition\_state』と、『pallet\_update\_flag』と、『pallet\_id』と、『composition\_object(1)～(m)』とから構成される。

『composition\_number』は、0から15までの数値を用いてDisplay Setにおけるグラフィックスアップデートを識別する。どのように識別するかというと、Epochの先頭から本PCSまでにグラフィックスアップデート  
30 が存在すれば、それらグラフィックスアップデートを経由する度に、イ



ンクリメントされるというルールでcomposition\_numberは設定される。

『composition\_state』は、本PCSから始まるDisplay Setが、Normal Caseであるか、Acquisition Pointであるか、Epoch Startであるかを  
5 示す。

『pallet\_update\_flag』は、本PCSにおいてPalletOnly Display Updateがなされているかどうかを示す。PalletOnly Display Updateとは、直前のパレットのみを新たなものに切り換えることでなされるアップデートをいう。本PCSでかかるアップデートがなされれば、本  
10 フィールドは" 1" に設定される。

『pallet\_id』は、PalletOnly Display Updateに用いられるべきパレットを示す。

『composition\_object(1)・・・(n)』は、このPCSが属するDisplay Setにおける画面構成を実現するための制御情報である。図8(b)の破  
15 線wd1は、任意のcomposition\_object(i)の内部構成をクローズアップしている。この破線wd1に示すように、composition\_object(i)は、  
『object\_id\_ref』、『window\_id\_ref』、『object\_cropped\_flag』、  
『object\_horizontal\_position』、『object\_vertical\_position』、  
『cropping\_rectangle情報(1)(2)・・・(n)』からなる。

20 『object\_id\_ref』は、グラフィクスオブジェクト識別子(object\_id)の参照値である。この参照値は、composition\_object(i)に対応する画面構成を実現するにあたって、用いるべきグラフィクスオブジェクトの識別子を意味する。

『window\_id\_ref』は、ウィンドウ識別子(window\_id)の参照値である。この参照値は、composition\_object(i)に対応する画面構成を実現するにあたって、どのウィンドウに、グラフィクスオブジェクトを表示させるべきかを示す。  
25

『object\_cropped\_flag』は、オブジェクトバッファにおいてクロップされたグラフィクスオブジェクトを表示するか、グラフィクスオブジェクトを非表示とするかを切り換えるフラグである。" 1" と設  
30

定された場合、オブジェクトバッファにおいてクロップされたグラフィックスオブジェクトが表示され、" 0" と設定された場合、グラフィックスオブジェクトは非表示となる。

5 『object\_horizontal\_position』は、グラフィックスプレーンにおけるグラフィックスオブジェクトの左上画素の水平位置を示す。

『object\_vertical\_position』は、グラフィックスプレーンにおける左上画素の垂直位置を示す。

『cropping\_rectangle情報(1)(2)……(n)』は、

10 『object\_cropped\_flag』が1に設定されている場合に有効となる情報要素である。破線wd2は、任意のcropping\_rectangle情報(i)の内部構成をクローズアップしている。この破線に示すようにcropping\_rectangle情報(i)は、

『object\_cropping\_horizontal\_position』、

『object\_cropping\_vertical\_address』、『object\_cropping\_width』、

15 『object\_cropping\_height』からなる。

『object\_cropping\_horizontal\_position』は、グラフィックスプレーンにおけるクロップ矩形の左上画素の水平位置を示す。クロップ矩形は、グラフィックスオブジェクトの一部を切り出すための枠であり、ETSI EN 300 743標準規格における" Region" に相当する。

20 『object\_cropping\_vertical\_address』は、グラフィックスプレーンにおけるクロップ矩形の左上画素の垂直位置を示す。

『object\_cropping\_width』は、グラフィックスプレーンにおけるクロップ矩形の横幅を示す。

25 『object\_cropping\_height』は、グラフィックスプレーンにおけるクロップ矩形の縦幅を示す。

以上がPCSのデータ構造である。続いてPCSの具体的な記述について説明する。この具体例は、図6に示した字幕の表示、つまり動画の再生進行に伴い、三回のグラフィックスプレーンへの書き込みで『ほんとは』『ウソだった』『ごめん』というように徐々に表示させるとい  
30 うものである。図9は、かかる字幕表示を実現するための記述例であ

る。本図におけるEpochは、DS1(Epoch Start)、DS2(Normal Case)、DS3(Normal Case)を有する。DS1は、字幕の表示枠となるwindowを定義するWDS、台詞『ほんとは ウソだった ごめん』を表すODS、1つ目のPCSを備える。DS2(Normal Case)は、2つ目のPCSを有する。

5 DS3(Normal Case)は3つ目のPCSを有する。

次に個々のPCSをどのように記述するかについて説明する。Display Setに属するWDS、PCSの記述例を図10～図12に示す。図10は、DS1におけるPCSの記述例を示す図である。

図10において、WDSのwindow\_horizontal\_position、  
10 window\_vertical\_positionは、グラフィックスプレーンにおけるウィンドウの左上座標LPlを、window\_width、window\_heightは、ウィンドウの表示枠の横幅、縦幅を示す。

図10におけるクロップ情報の  
object\_cropping\_horizontal\_position、object\_cropping\_vertical\_  
15 positionは、オブジェクトバッファにおけるグラフィクスオブジェクトの左上座標を原点とした座標系においてクロップ範囲の基準ST1を示している。そして基準点からobject\_cropping\_width、object\_cropping\_heightに示される範囲(図中の太枠部分)がクロップ範囲になる。クロップされたグラフィクスオブジェクトは、グラフィックスプレーンの座標系において  
20 object\_horizontal\_position、object\_vertical\_positionを基準点(左上)とした破線の範囲cp1に配置される。こうすることにより、『本当は』がグラフィックスプレーンにおけるウィンドウ内に書き込まれる。これにより字幕『本当は』は動画像と合成され表示される。

25 図11(a)は、DS2におけるPCSの記述例を示す図である。本図におけるWDSの記述は、図10と同じなので説明を省略する。クロップ情報の記述は、図10と異なる。図11(a)におけるクロップ情報の

object\_cropping\_horizontal\_position、object\_cropping\_vertical\_  
30 position、は、オブジェクトバッファ上の字幕『本当はウソだった。

ごめん』のうち、『ウソだった』の左上座標を示し、  
object\_cropping\_height, object\_cropping\_widthは、『ウソだった』  
の横幅、縦幅を示す。こうすることにより、『ウソだった』がグラフィックスプレーンにおけるウィンドウ内に書き込まれる。これにより  
5 字幕『ウソだった』は動画像と合成され表示される。

図 1 1 (b) は、DS3におけるPCSの記述例を示す図である。本図におけるWDSの記述は、図 1 0 と同じなので説明を省略する。クロップ情報の記述は、図 1 0 と異なる。図 1 1 (b) におけるクロップ情報の

10 object\_cropping\_horizontal\_position, object\_cropping\_vertical\_position, は、オブジェクトバッファ上の字幕『本当はウソだった。ごめん』のうち、『ごめん』の左上座標を示し、  
object\_cropping\_height, object\_cropping\_widthは、『ごめん』の横幅、縦幅を示す。こうすることにより、『ごめん』がグラフィックス  
15 プレーンにおけるウィンドウ内に書き込まれる。これにより字幕『ごめん』は動画像と合成され表示される。 図 1 3 は、図 1 0 ~ 図 1 2 に示すようなグラフィクスアップデートを実現するにあたっての、オブジェクトバッファにおけるメモリ空間を示す図である。本図に示すようにオブジェクトバッファは、縦幅、横幅、位置が固定された格納  
20 領域A, B, C, Dが4つあり、このうち格納領域Aに、図 1 1 に示した字幕が格納される。これら4つの格納領域A, B, C, Dのそれぞれには、格納されているグラフィクスオブジェクトに対応するobject\_idにより識別される。つまりobject\_id=1により格納領域Aが、object\_id=2により格納領域Bが、object\_id=3により格納領域Cがそれぞれ識別されるのである。グラフィックスプレーンへの転送効率を維持するため、これ  
25 ら格納領域の縦幅、横幅は終始固定されている。同じobject\_idをもったグラフィクスオブジェクトがデコードにより新たに得られれば、それらの格納領域はその新たに得られたグラフィクスオブジェクトにより上書きされる。例えば、図 1 0 に示した字幕と同じ位置に、同  
30 じ大きさで字幕を表示させたい場合、後続するDisplay Setにおいて、

同じobject\_idを付加したODSを設ければよい。このように同じobject\_idを付加するだけで、オブジェクトバッファ上に存在するグラフィックスオブジェクトは、新たなグラフィックスオブジェクトで上書きされ、その新たなグラフィックスオブジェクトは、先のグラフィックスオブジェクトと同じ大きさ・同じ位置で表示されることになる。

表示効果実現にあたっての制約について説明する。なめらかな表示効果の実現には、ウィンドウクリアと、ウィンドウ再描画とが必要になる。ウィンドウクリアと、ウィンドウ再描画とをビデオフレームの時間間隔で実現する場合、オブジェクトバッファと、グラフィックスプレーンとの間の転送レートはどれだけ必要であろうか。

ここでウィンドウをどれだけの大きさとするかの制限について検討する。オブジェクトバッファ・グラフィックスプレーン間の転送レートを $R_c$ とすると、ワーストケースでは、この転送レート $R_c$ でウィンドウクリアと、ウィンドウ再描画とを行わねばならない。そうするとウィンドウクリア、ウィンドウ再描画のそれぞれを $R_c$ の半分の転送レート( $R_c/2$ )で実現せねばならない。

これらウィンドウクリアーウィンドウ再描画をビデオフレームと同期させるには、

$$\text{ウィンドウサイズ} \times \text{フレームレート} \leq R_c/2$$

を満たす必要がある。このフレームレートが29.97であるなら、

$$R_c \text{は、ウィンドウサイズ} \times 2 \times 29.97 \text{になる。}$$

字幕の表示にあたっては、グラフィックスプレーン全体に対し、最低でも25%~33%程度の大きさが必要となる。ここでグラフィックスプレーンの総画素数は $1920 \times 1080$ であり、一画素当たりのインデックスのビット長を8ビットとすると、グラフィックスプレーンの総容量は2Mバイト( $\equiv 1920 \times 1080 \times 8$ )になる。

この総容量の1/4をウィンドウサイズとすると、500kバイト( $= 2\text{Mバイト}/4$ )になる。これを上述した式に代入して $R_c$ を導けば、 $R_c$ は256Mbps( $500\text{Kバイト} \times 2 \times 29.97$ )と算出することができる。

この25%~33%という大きさであれば、256Mbpsという転送レートで

字幕の表示を行っている限り、如何なる表示効果を実現する場合であっても、動画との同期を維持することができる。

仮に、ウィンドウクリア及び再描画のレートがビデオフレームのフレームレートの1/2, 1/4でよいなら、Rcがたとえ同じであってもウィンドウサイズを2倍, 4倍にすることができる。以上がwindowの大きさ  
5 についての説明である。続いて、windowの位置、範囲について説明する。上述したように、Epochにおいてウィンドウの位置、範囲は一貫している。

Epochにおいてウィンドウの位置、範囲を一貫させておくのは以下の理由による。ウィンドウの位置・範囲を変えれば、グラフィックス  
10 プレーンに対する書込先アドレスを変えねばならず、オーバーヘッドが発生するので、かかるオーバーヘッドによりオブジェクトバッファからグラフィックスプレーンへの転送レートが低下するからである。

ウィンドウには、グラフィクスオブジェクトの個数が制限されている。この個数制限は、デコードされたグラフィクスオブジェクトの転  
15 送にあたってのオーバーヘッドを低減する目的で設けられている。ここでのオーバーヘッドは、グラフィクスオブジェクトのエッジ部分のアドレスを設定する際に発生する。そうすれば、エッジの部分が多く存在する程、オーバーヘッドの発生回数が多くなる。

ウィンドウにおけるグラフィクスオブジェクトの数に制限がない  
20 と、グラフィクス転送にあたって発生するオーバーヘッド数が未知数になり、転送負荷の増減が激しくなる。一方、ウィンドウにおけるグラフィクスの個数が2つまでであると、最悪4つのオーバーヘッドが発生すると見込んで転送レートを設定すればよいので、ミニマムスタン  
25 ダードたる転送レートを数値化し易くなる。

以上がウィンドウについての説明である。続いてこれらPCS、ODSを  
有したDisplay Setが、AVClipの再生時間軸上にどのように割り当て  
られるかについて説明する。Epochは、再生時間軸上においてメモリ  
管理が連続する期間であり、Epochは1つ以上のDisplay Setから構成  
30 されるので、Display SetをどうやってAVClipの再生時間軸に割り当

てるかが問題になる。ここでAVClipの再生時間軸とは、AVClipに多重されたビデオストリームを構成する個々のピクチャデータのデコードタイミング、再生タイミングを規定するために想定される時間軸をいう。この再生時間軸においてデコードタイミング、再生タイミングは、90KHzの時間精度で表現される。Display Set内のPCS、ODSに付加されたDTS、PTSは、この再生時間軸において同期制御を実現すべきタイミングを示す。このPCS、ODSに付加されたDTS、PTSを用いて同期制御を行うことが、再生時間軸へのDisplay Setの割り当てである。

10 5 10 15 20 25 30  
まず、ODSに付加されたDTS、PTSにより、どのような同期制御がなされるかについて説明する。

DTSは、ODSのデコードを開始すべき時間を90KHzの時間精度で示しており、PTSはデコード終了時刻を示す。

ODSのデコードは、瞬時には完了せず、時間的な長さをもっている。このデコード期間の開始点・終了点を明らかにしたいとの要望から、ODSについてのDTS、PTSはデコード開始時刻、デコード終了時刻を示している。

PTSの値はデッドラインであるので、PTSに示される時刻までにODSのデコードがなされて、非圧縮状態のグラフィックオブジェクトが、再生装置上のオブジェクトバッファに得られなければならない。

20 DSnに属する任意のODSjのデコード開始時刻は、90KHzの時間精度でDTS(DSn[ODSj])に示されるので、これにデコードを要する最長時間を加えた時刻が、Display SetのODSjのデコード終了時刻になる。

ODSjのサイズを"SIZE(DSn[ODSj])"、ODSのデコードレートを"Rd"とすると、デコードに要する最長時間(秒)は、"SIZE(DSn[ODSj])/Rd"になる。ここで"//"は、小数点以下切り上げの割り算を示す演算子である。

この最長時間を90KHzの時間精度に変換し、ODSjのDTSに加算することにより、PTSで示されるべきデコード終了時刻(90KHz)は算出される。

DSnに属するODSjのPTSを、数式で表すと、以下の式のようにになる。

$$PTS(DS[ODSj]) = DTS(DSn[ODSj]) + 90,000 \times (SIZE(DSn[ODSj]) / Rd)$$

そして互いに隣接する2つのODS(ODSj, ODSj+1)との間では、以下の関係を満たす必要がある。

$$PTS(DSn[ODSj]) \leq DTS(DSn[ODSj+1])$$

- 5      またDSnに属するENDは、DSnの終わりを示すものだから、DSnに属する最後のODS(ODSlast)のデコード終了時刻を示せばよい。このデコード終了時刻は、ODS2(ODSlast)のPTS(PTS(DSn[ODSlast]))に示されているので、ENDのPTSは、以下の式に示される値に設定される。

$$PTS(DSn[END]) = PTS(DSn[ODSlast])$$

- 10      続いてPCSのDTS、PTSの設定について説明する。

PCSのDTS値は、DSnにおける最初のODS(ODS1)のデコード開始時点か、それ以前の時点を示す。何故ならPCSは、DSnにおける最初のODS(ODS1)のデコード開始時点(DTS(DSn[ODS1]))、及び、DSnにおける最初のPDS(PDS1)が有効になる時点(PTS(DSn[PDS1]))と同時か、又はそれ以前に、再生装置上のバッファにロードされねばならないからである。

15      よって以下の式の関係を満たす値に、設定されねばならない。

$$DTS(DSn[PCS]) \leq DTS(DSn[ODS1])$$

$$DTS(DSn[PCS]) \leq PTS(DSn[PDS1])$$

そしてDSnにおけるPCSのPTSは、以下の数式から算出される。

$$20 \quad PTS(DSn[PCS]) \geq DTS(DSn[PCS]) + decodeduration(DSn)$$

ここでdecodeduration(DSn)は、PCSのアップデートに用いられる全グラフィクスオブジェクトのデコード及び表示に要する時間である。このデコード時間は、固定値ではない。しかし各再生装置の状態や再生装置の実装により変動するものでもない。本DSn.PCSnの画面合成に

- 25      用いられるオブジェクトをDSn.PCSn.OBJ[j]とした場合、

decodeduration(DSn)は、ウィンドウクリアに要する時間(i)、

DSn.PCSn.OBJのデコード期間(ii)、DSn.PCSn.OBJの書き込みに要する時間(iii)により変動を受ける値になる。Rd, Rcさえ予め定められていれば、どのような実装の再生装置であっても、同じ値になる。そのため

- 30      オーサリング時にあたっては、これらの期間の長さを算出して、こ



の値からPTSを計算している。

decode\_durationの計算は、図14のプログラムに基づき行われる。  
また図15、図16(a)(b)は、このプログラムのアルゴリズム  
を図式化したフローチャートである。以降、これらの図を参照しながら  
5 decode\_durationの算出について説明する。図15のフローチャート  
において、先ず初めに、PLANEINITIALIZATINTIME関数を呼び出し、  
戻り値をdecode\_durationに加算する(図15のステップS1)。

PLANEINITIALIZATINTIME関数(図16(a))は、Display Setを表示  
するにあたってグラフィックスプレーンの初期化に要する時間を求  
10 める関数を呼出するための関数であり、図15のステップS1では、  
DSn, DSn.PCS.OBJ[0], decode\_durtationを引数に設定して、この関数  
を呼び出す。

続いて図16(a)を参照しながらPLANEINITIALIZATINTIME関数に  
ついて説明する。図16(a)においてinitialize\_durationとは、  
15 PLANEINITIALIZATINTIME関数の戻り値を示す変数である。

図16のステップS2は、DSnのPCSにおけるcomposition\_stateが  
Epoch Startかどうかにより、処理を切り換えるif文である。もし  
composition\_stateがEpoch Startであるなら(図14の  
DSn.PCS.composition\_state==EPOCH\_START、ステップS2=Yes)、グ  
20 ラフィックスプレーンのクリアに要する時間をinitialize\_duration  
に設定する(ステップS3)。

オブジェクトバッファグラフィックスプレーン間の転送レート  
Rcを上述したように256,000,000とし、グラフィックスプレーンの総  
サイズをvideo\_width\*video\_heightとすると、クリアに要する時間  
25 (秒)は、「video\_width\*video\_height//256,000,000」になる。これ  
に、90,000Hzを乗じPTSの時間精度で表現すれば、グラフィックスプ  
レーンのクリアに要する時間は「90000×  
video\_width\*video\_height//256,000,000」になる。この時間を、  
initialize\_durationに加えて、initialize\_durationを戻り値として  
30 リターンする。

もし composition\_state が Epoch Start でないなら (ステップ S 2 = No)、WDS にて定義される window[i] のクリアに要する時間を initialize\_duration に加えるという処理を全ての window について繰り返す (ステップ S 4)。オブジェクトバッファグラフィックスプレーン間の転送レート Rc を上述したように 256,000,000 とし、WDS に属するウィンドウ [i] の総サイズを  $\Sigma \text{SIZE}(\text{WDS.WIN}[i])$  とすると、クリアに要する時間 (秒) は、「 $\Sigma \text{SIZE}(\text{WDS.WIN}[i]) / 256,000,000$ 」になる。これに、90,000Hz を乗じ PTS の時間精度で表現すれば、WDS に属する全ウィンドウのクリアに要する時間は「 $90000 \times \Sigma \text{SIZE}(\text{WDS.WIN}[i]) / 256,000,000$ 」になる。この時間を、initialize\_duration に加えて、initialize\_duration を戻り値としてリターンする。以上が PLANEINITIALIZATINTIME 関数である。

図 15 のステップ S 5 は、DSn に属するグラフィックスオブジェクトの数が 2 であるか、1 であるかで処理を切り換えるものであり (図 14 の if (DSn.PCS.num\_of\_object==2), if (DSn.PCS.num\_of\_object==1))、もし "1" であるなら (ステップ S 5 = 1)、グラフィックスオブジェクトのデコードを行うための待ち時間を decode\_duration に加える (ステップ S 6)。この待ち時間の算出は、WAIT 関数の呼出 (図 14 の decode\_duration += WAIT (DSn, DSn.PCS.OBJ[0], decode\_duration)) で実現される。この関数呼出にあたっての引数は DSn, DSn.PCS.OBJ[0], decode\_duration であり、待ち時間を示す wait\_duration が戻り値として返される。

図 16 (b) は、WAIT 関数の処理手順を示すフローチャートである。

この WAIT 関数において current\_duration とは、呼出元の decode\_duration が設定される。object\_define\_ready\_time は、Display Set のグラフィックスオブジェクト [i] の PTS が設定される変数である。

current\_time とは、DSn の PCS の DTS に、current\_duration を足した値が設定される変数である。この current\_time より object\_define\_ready\_time が大きい場合 (ステップ S 7 が Yes、

if(current\_time < object\_define\_ready\_time)), 戻り値たる wait\_durationは、object\_define\_ready\_timeとcurrent\_timeとの差分が設定されることになる(ステップS 8、wait\_duration += object\_define\_ready\_time - current\_time)。以上がWait関数である。

- 5 ステップS 6においてdecode\_durationには、このwait関数の戻り値と、OBJ[0]をwindowに描画するのに必要な時間を足し合わせた時間(90,000\*(SIZE(DSn.WDS.WIN[0]))//256,000,000)が設定されることになる(ステップS 9)。

- 10 以上はDSnのグラフィクスオブジェクトが1つである場合の処理である。図15のステップS 5は、グラフィクスオブジェクトの数が2であるかどうかを判定している。DSnのグラフィクスオブジェクトが2以上であれば(図14のif(DSn.PCS.num\_of\_object==2))、DSn, DSn.PCS.OBJ[0], decode\_durationを引数として、wait関数を呼び出し、その戻り値をdecode\_durationに加算する(ステップS 10)。

- 15 続くステップS 11は、DSnのOBJ[0]が属するwindowが、グラフィクスオブジェクト[1]が属するwindowと同一かどうかの判定であり(if(DSn.PCS.OBJ[0].window\_id == DSn.PCS.OBJ[1].window\_id)、もし同一であるなら、DSn, DSn.PCS.OBJ[1], decode\_durationを引数にしてwait関数を呼び出し、その戻り値wait\_durationを、  
20 decode\_durationに加算する(ステップS 12)。OBJ[0]が属するwindowの再描画に必要な時間(90,000\*(SIZE(DSn.WDS.OBJ[0].window\_id)//256,000,000)をdecode\_durationに加える(ステップS 13)。

- もし属するwindowが異なるなら(ステップS 11で"異なる")、  
25 OBJ[0]が属するwindowの再描画に必要な時間(90,000\*(SIZE(DSn.WDS.OBJ[0].window\_id)//256,000,000)をdecode\_durationに加える(ステップS 15)。  
DSn, DSn.PCS.OBJ[1], decode\_durationを引数にしてwait関数を呼び出し、その戻り値wait\_durationを、decode\_durationに加算する(ステップS 16)。その後、OBJ[1]が属するwindowの再描画に必要な時  
30

間  $(90,000 * (\text{SIZE}(\text{DSn.WDS.OBJ}[1].\text{window\_id}) // 256,000,000))$  を  
decode\_durationに加える(ステップ S 17)。

以上のアルゴリズムにより、Decode\_Durationは算出される。この  
PCSのPTSが、具体的にどのように設定されるかについて説明する。

- 5 図 17 (a) は、1つのwindowに1つのODSが存在するケースを想定  
した図である。図 17 (b)・(c) は、図 14 で引用した各数値の時  
間的な前後関係を示すタイミングチャートである。本チャートには3  
つの段があり、これらの段のうち、“グラフィックスプレーンアクセ  
ス”の段、“ODSデコード”の段は、再生時にパラレルになされる2つ  
10 の処理を示す。上述したアルゴリズムは、これらの2つの処理のパラ  
レル実行を想定している。

グラフィックスプレーンアクセスは、クリア期間(1)、書き込み期  
間(3)からなる。このクリア期間(1)は、グラフィックスプレーン全体  
のクリアに要する期間  $(90,000 * (\text{グラフィックスプレーンのサイズ}$   
15  $// 256,000,000))$ 、グラフィックスプレーンにおける全windowのクリ  
アに要する時間  $(\sum (90,000 * (\text{window}[i] \text{のサイズ} // 256,000,000)))$   
のどちらかを意味する。

書き込み期間(3)は、window全体描画に要する期間  $(90,000 * (\text{windowサイズ} // 256,000,000))$  を意味する。

- 20 一方、デコード期間(2)は、ODSのDTSからPTSまでに示される期間を  
意味する。これらクリア期間(1)～書き込み期間(3)は、クリアすべ  
き範囲、デコードすべきODSのサイズ、グラフィックスプレーンに書  
き込むべきグラフィクスオブジェクトのサイズにより変化し得る。本  
図では、説明の簡略化のため、ODSのデコード期間(2)の始点は、クリ  
ア期間(1)の始点と同一であると仮定している。  
25

図 17 (b) はデコード期間(2)が長くなるケースであり、  
Decode\_Durationはデコード期間(2)＋書き込み期間(3)になる。

- 図 17 (c) は、クリア期間(1)が長くなるケースであり、  
Decode\_Durationはクリア期間(1)＋書き込み期間(3)の期間が  
30 Decode\_Durationになる。

図18(a)～(c)は、1つのwindowに2つのODSが存在するケースを想定した図である。本図(b)(c)におけるデコード期間(2)は、2つのグラフィックスのデコードに要する期間の総和を意味する。グラフィックス書込期間(3)も、2つのグラフィックスをグラフィックスプレーンに書き込む期間の総和を意味する。ODSが2つになっているものの、図17と同様に考えればDecode\_Durationを算出することができる。2つのODSをデコードするためのデコード期間(2)が長い場合は、図18(b)に示すようにDecode\_Durationはデコード期間(2)＋書き込み期間(3)に算出されることになる。

- 10 クリア期間(1)が長い場合は、図18(c)に示すように、Decode\_Durationはクリア期間(1)＋書き込み期間(3)となる。

- 図19(a)は、2つのwindowのそれぞれに、ODSが1つずつ存在するケースを想定している。この場合でもクリア期間(1)が、2つのODSをデコードするための総デコード期間(2)より長い場合、
- 15 Decode\_Durationはクリア期間(1)＋書き込み期間(3)になる。問題は、クリア期間(1)がデコード期間(2)より短くなるケースである。この場合デコード期間(2)の経過を待たずに、1つ目のwindowへの書き込みは可能になる。そのためクリア期間(1)＋書き込み期間(3)、デコード期間(2)＋書き込み期間(3)の長さにはならない。ここで1つ目のODSのデ
- 20 コードに要する期間を書込期間(31)、2つ目のODSのデコードに要する期間を書込期間(32)とする。図19(b)は、デコード期間(2)がクリア期間(1)＋書込期間(31)より長くなるケースを示す。この場合Decode\_Durationは、デコード期間(2)＋書込期間(32)になる。

- 図19(c)は、クリア期間(1)＋書込期間(31)がデコード期間(2)より長くなるケースを示す。この場合Decode\_Durationはクリア期間(1)＋書込期間(31)＋書込期間(32)になる。
- 25

- グラフィックスプレーンのサイズは、プレーヤモデルから予め判明しており、またwindowのサイズ、ODSのサイズ、個数もオーサリングの段階で判明しているので、これらからDecode\_Durationがクリア期間(1)＋書き込み期間(3)、デコード期間(2)＋書き込み期間(3)、デコ
- 30

ード期間(2)+書込期間(32)、クリア期間(1)+書込期間(31)+書込期間(32)のどれかになる。こうしたDecode\_Duration算出を基にPCSのPTSを設定すれば、ピクチャデータとの同期表示を高い時間精度で実現することができる。このような高精度な同期制御は、windowを定義し、クリア・再描画する範囲を、このwindowに限定することで成り立っている。このwindowの概念を導入したことの意義は大きい。

続いてDS<sub>n</sub>におけるWDSのDTS、PTSの設定について説明する。WDSのDTSは、以下の数式を満たすように設定すればよい。

$$10 \quad DTS(DS_n[WDS]) \geq DTS(DS_n[PCS])$$

一方、DS<sub>n</sub>におけるWDSのPTSは、グラフィックスプレーンに対する書き込みを開始すべきデッドラインを示す。グラフィックスプレーンへの書き込みは、ウィンドウだけで足りるので、PCSのPTSに示される時刻から、WDSの書き込みに要する期間を差し引けば、グラフィックスプレーンへの書き込みを開始すべき時刻は定まる。WDSの総サイズを $\sum SIZE(WDS.WIN[i])$ とすれば、このクリア及び再描画に要する時間は、『 $\sum SIZE(WDS.WIN[i])/256,000,000$ 』になる。そして、これを90.000KHzの時間精度で表現すると『 $90000 \times \sum SIZE(WDS.WIN[i])/256,000,000$ 』になる。

20 故に以下の数式から、WDSのPTSを算出すればよい。

$$PTS(DS_n[WDS]) = PTS(DS_n[PCS]) - 90000 \times \sum SIZE(WDS.WIN[i])/256,000,000$$

このWDSに示されるPTSはデッドラインなので、これより早い時点からグラフィックスプレーンの書き込みを開始してもよい。つまり図19に示すように、2つのウィンドウのうち、1つのウィンドウに表示させるべきODSのデコードが完了したなら、その時点からデコードにより得られたグラフィックスオブジェクトの書き込みを開始してもよい。

30 このようにWDSに付加されたDTS、PTSを用いてAVC1ipの再生時間軸の任意の時点に、ウィンドウを割り当てることができる。

以上がPCS、WDSのDTS、PTSの説明である。

続いてPCSによる前方参照について説明する。前方参照とは、前もって再生装置のメモリに読み込まれたグラフィクスオブジェクトを参照した再生制御をいう。図20は、前方参照を行うPCSを描いた図である。本図では、DS<sub>n</sub>、DS<sub>n+1</sub>、DS<sub>n+2</sub>という3つのDisplay Setが描かれている。この中でDS<sub>n+1</sub>、DS<sub>n+2</sub>に属するPCSが「前方参照を行うPCS」である。本図の矢印yy1,yy2は、PCSにおける前方参照を象徴的に示す。かかる前方参照において参照先になっているグラフィクスオブジェクトは、DS<sub>n</sub>内のODS#1～#vにより定義されるグラフィクスオブジェクトである。ここでDS<sub>n</sub>は、ODS#1～#u、ODS#u+1～#vという複数のグラフィクスオブジェクトを含む。このうちDS<sub>n</sub>内のPCSにより参照されているのは、ODS#1～#uであり、ODS#u+1～#vは参照されていない。つまりDS<sub>n</sub>内の非参照グラフィクスオブジェクトが前方参照されているのである。

以上のような前方参照では、PCSにより参照されているグラフィクスオブジェクトが前もって再生装置に読み込まれているので、同じDisplay Setに属するODSのデコードを待つ必要が無い。PCSを新たに送り込むだけで、そのPCSに基づく表示制御を再生装置に実行させることができる。PCSの読み込みに即応した表示制御が可能であるため、既に表示されているグラフィクスを、新たなグラフィクスを用いて更新するというグラフィクス更新を短い時間間隔で実行することができる。そして短い時間間隔でのグラフィクス更新を繰り返せば、動画像の再生進行と共にグラフィクスを滑らかに動かすという表示制御が可能になる。前方参照型のPCSにおける2つの記述例について説明する。

先ず初めに、画面上でグラフィクスが動きまわるような表示効果を実現するにあたってのPCSの記述について説明する。図21(a)は、かかる表示効果を実現するEpochを示す図である。本Epochは、9つのDisplay Set(DS0～DS8)からなり、先頭のDS0は、PCS、PDS、ODSを含む。DS0内のODSは、図22(a)に示すような字幕「心が揺れる」を

定義するものであり、object\_id=1が付されている。後続するDS1～DS8は、PCSのみを含む。

図21(b)は、DS1～DS8に含まれるPCSの内容と、PTS値とを示す図である。本図に示すようにDS1～DS8に属するPCS内の内容は、

- 5 object\_id\_ref=1のグラフィクスオブジェクトを座標  
(x1,y1)(x2,y2)(x3,y3)・・・(x8,y8)に表示するよう、  
object\_horizontal\_position,object\_vertivcal\_positionの位置決  
めがなされている。これら座標(x1,y1)(x2,y2)(x3,y3)・・・(x8,y8)が、  
ウィンドウの座標系のどこに存在するかを、具体的に示したのが図2  
10 2(b)である。図22(b)において座標(x1,y1)(x2,y2)(x3,y3)・・・  
(x8,y8)は、波形を表す曲線上に存在していることがわかる。これら  
(x1,y1)(x2,y2)(x3,y3)・・・(x8,y8)という座標値は、WDSにて定義され  
るウィンドウ内に、グラフィクスが収まるように規定されている。ウ  
ィンドウ内に、グラフィクスが収まらなと、ビデオストリームにお  
15 ける動画表示との同期を確立することができないからである。

図21(b)におけるPCSのPTS値は、座標(x1,y1)(x2,y2)(x3,y3)・・・  
(x8,y8)に対する表示を、再生時間軸上の時点t1,t2,t3・・・t8のタイ  
ミングで命じるものである。

- 以上のように、PCSの内容が規定されたDS0～DS8を順次再生装置に  
20 読み込んでゆくことにより、グラフィクスは、図23のような表示効  
果をなす。動画ストリームの再生時間軸における時点t1において、再  
生装置にはDS1が読み込まれるので、ウィンドウ内の座標(x1,y1)にグ  
ラフィクスが表示される。時点t4において再生装置にはDS4が読み込  
まれるので座標(x4,y4)にグラフィクスが表示されることになる。一  
25 方時点t6では、DS6が読み込まれるので座標(x6,y6)に、時点t8ではDS8  
が読み込まれるので座標(x8,y8)にグラフィクスが表示されることにな  
る。これらの座標(x1,y1),(x4,y4),(x6,y6),(x8,y8)は、波形の曲  
線上に存在するので、かかる座標に表示位置の位置決めがなされた  
PCSにより、グラフィクスは、図24に示すように波形の軌道を描く  
30 ことになる。



以上のように

object\_horizontal\_position, object\_vertivcal\_positionが規定されたPCSを含むDisplay Setを順次再生装置に読み込ませれば、再生装置はBD-ROMから読み出されてくるDS1、DS2、DS3内のPCSに従い、既に  
5 オブジェクトバッファに格納されているグラフィクスオブジェクトを、表示させることができる。制御情報の送り込みのみで、グラフィクスの位置を変化させることができるので、グラフィクスを目まぐるしく動かすという制御に適している。

個々のDisplay Setの読み出し時において、それまで参照していた  
10 PCSを、新たなPCSで上書きするとの動作を行えば、たとえ読み出すべきPCSが何十個、何百個あったとしても、そのうち最も新しいもののみ、再生装置に格納しておけばよい。滑らかな動きを実現するため、表示座標を示すPCSが、何十個、何百個必要であっても、そのうち最新のもののみ、再生装置に常駐させておけばよいので、再生装置にお  
15 けるメモリ占有量を必要最低限にすることができる。

続いてPCSの第2の記述例としてPallet Only Updateについて説明する。図25は、Pallet Only Updateを実現する一連のDisplay Setを示す図である。本図の第1段目は、Pallet Only Updateを実現する一連のDisplay Set(DS0、DS1、DS2、DS3)を示す。このうちDS0は、PCS  
20 と、3つのPDS0,1,2,3と、ODSとを含む。以降のDS2、DS3、DS4は、PCSのみからなる。

DS0内のODSは、引き出し線hv1に示すように、文字列「忘れない」を定義するものである。これら文字「忘」「れ」「な」「い」は、それぞれ複数のRun Length(図中の白抜きの四角形)から構成される。このうち文字「忘」を構成するRun Lengthは、PixelCode0をもっているものとする。文字「れ」、「な」、「い」を構成するRun Lengthは、PixelCode1, PixelCode2, PixelCode3をもっているものとする。  
25

図26(a)は、DS0内のPDS及び各Display SetのPCSがどのような内容であるかを示す図である。

30 4つのPDSのうち、1つ目のPDS0は(第1段目)、pallet\_id=0が付加さ

れており、PixelCode1に赤色を、PixelCode2以降に白色を割り当てるものである。

2つ目のPDS1は(第2段目)、pallet\_id=1が付加されており、1,2のPixelCodeに赤色を、3以降のPixelCodeに白色を割り当てるものである。

3つ目のPDS2は(第3段目)、pallet\_id=2が付加されており、1,2,3のPixelCodeに赤色を、4のPixelCodeに白色を割り当てるものである。

4つ目のPDS3は(第4段目)、pallet\_id=3が付加されており、1~4のPixelCodeに赤色を割り当てるものである。

10 図26(b)は、DS0~DS3内のPCSを示す図である。

本図においてDS0内のPCSは(第1段目)、object\_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そして

Pallet\_only\_updateが0に設定され、Pallet\_idが0に設定されている。これによりDS0内のODSは、Pallet\_id=0で指示されるPDSを用いて、表示されることになる。

DS1内のPCSは(第2段目)、object\_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet\_only\_updateが1に設定され、Pallet\_idが1に設定されている。これは、DS1の読み込み時において、Pallet\_id=1で指示されるPDSを用いて、グラフィクス表示の更新を行う旨を示す。

DS2内のPCSは(第3段目)、object\_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet\_only\_updateが1に設定され、Pallet\_idが2に設定されている。これは、DS2の読み込み時において、Pallet\_id=2で指示されるPDSを用いて、グラフィクス表示の更新を行う旨を示す。]

DS3内のPCSは(第4段目)、object\_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet\_only\_updateが1に設定され、Pallet\_idが3に設定されている。これは、DS3の読み込み時において、Pallet\_id=3で指示されるPDSを用いて、グラフィクス表示の更新を行う旨を示す。DS2~DS3に属するPCSは、以上のように

設定されているので、DS0に属するグラフィクスオブジェクトは、それぞれ別々のパレットデータを用いて、更新されてゆくことになる。

図27は、これら4つのDisplay Setが読み込ませることで実現される、表示効果を現した図である。DS0に属するグラフィクスオブジェクトが再生装置に投入された段階で、字幕「忘れない」が表示され、それを構成する文字のうち、文字「忘」が赤色になる。次にDS1を投入した段階において、字幕「忘れない」を構成する文字のうち、文字「忘」、「れ」が赤色に、DS2を投入した段階で、文字「忘」、「れ」、「な」が赤色になる。そしてDS3を投入した段階で、全ての文字が赤色になる。Display Setが投入される毎に、字幕を構成する文字は、左に位置するものから順に、白色から赤色に変わってゆくので、音声の進行と共に、文字の色彩を代えてゆく表示効果を実現することができる。

これまでに説明したDisplay Set(PCS, WDS, PDS, ODS)のデータ構造は、プログラミング言語で記述されたクラス構造体のインスタンスであり、オーサリングを行う制作者は、Blu-ray Disc Read Only Formatに規定された構文に従い、クラス構造体を記述することで、BD-ROM上のこれらのデータ構造を得ることができる。以上が本発明に係る記録媒体の実施形態である。続いて本発明に係る再生装置の実施形態について説明する。図28は、本発明に係る再生装置の内部構成を示す図である。本発明に係る再生装置は、本図に示す内部に基づき、工業的に生産される。本発明に係る再生装置は、主としてシステムLSIと、ドライブ装置、マイコンシステムという3つのパーツからなり、これらのパーツを装置のキャビネット及び基板に実装することで工業的に生産することができる。システムLSIは、再生装置の機能を果たす様々な処理部を集積した集積回路である。こうして生産される再生装置は、BDドライブ1、Read Buffer2、PIDフィルタ3、Transport Buffer4 a, b, c、周辺回路4 d、ビデオデコーダ5、ビデオプレーン6、オーディオデコーダ7、グラフィクスプレーン8、CLUT部9、加算器10、グラフィクスデコーダ12、Coded Data Buffer13、周辺回路13 a、Stream Graphics Processor14、Object Buffer15、Composition

Buffer 1 6、Graphical Controller 1 7 から構成される。

BD-ROMドライブ 1 は、BD-ROMのローディング／リード／イジェクトを行い、BD-ROMに対するアクセスを実行する。

Read Buffer 2 は、FIFOメモリであり、BD-ROMから読み出されたTS  
5 パケットが先入れ先出し式に格納される。

PIDフィルタ 3 は、Read Buffer 2 から出力される複数TSパケットに  
対してフィルタリングを施す。PIDフィルタ 3 によるフィルタリング  
は、TSパケットのうち、所望のPIDをもつもののみをTransport Buffer  
4 a, b, cに書き込むことでなされる。PIDフィルタ 3 によるフィルタリ  
10 ングでは、バッファリングは必要ではない。従って、PIDフィルタ 3  
に入力されたTSパケットは、時間遅延なく、Transport Buffer 4 a, b, c  
に書き込まれる。

Transport Buffer 4 a, b, cは、PIDフィルタ 3 から出力されたTSパケ  
ットを先入れ先出し式に格納しておくメモリである。このTransport  
15 Buffer 4 aからTSパケットが取り出される速度を速度Rxとする

周辺回路 4 dは、Transport Buffer 4 a, b, cから読み出されたTSパケ  
ットを、機能セグメントに変換する処理を行うワイアロジックである。  
変換により得られた機能セグメントはCoded Data Buffer 1 3 に格納  
される。

20 ビデオデコーダ 5 は、PIDフィルタ 3 から出力された複数TSパケッ  
トを復号して非圧縮形式のピクチャを得てビデオプレーン 6 に書き  
込む。

ビデオプレーン 6 は、動画用のプレーンメモリである。

オーディオデコーダ 7 は、PIDフィルタ 3 から出力されたTSパケッ  
25 トを復号して、非圧縮形式のオーディオデータを出力する。

グラフィクスプレーン 8 は、一画面分の領域をもったプレーンメモ  
リであり、一画面分の非圧縮グラフィクスを格納することができる。

CLUT部 9 は、グラフィクスプレーン 8 に格納された非圧縮グラフィ  
クスにおけるインデックスカラーを、PDSに示されるY, Cr, Cb値に基づ  
30 き変換する。

加算器 10 は、CLUT部 9 により色変換された非圧縮グラフィクスに、PDSに示されるT値(透過率)を乗じて、ビデオプレーン 6 に格納された非圧縮状態のピクチャデータと画素毎に加算し、合成画像を得て出力する。

- 5      グラフィクスデコーダ 12 は、グラフィクスストリームをデコードして、非圧縮グラフィクスを得て、これをグラフィクスオブジェクトとしてグラフィクスプレーン 8 に書き込む。グラフィクスストリームのデコードにより、字幕やメニューが画面上に現れることになる。

- 10      グラフィクスデコーダ 12 によるパイプラインは、DSnに属するグラフィクスオブジェクトをObject Buffer 15 に書き込む処理、DSn+1 に属するグラフィクスオブジェクトをObject Buffer 15 から読み出す処理を同時に実行することでパイプラインは実行される。

- 15      このグラフィクスデコーダ 12 は、Coded Data Buffer 13、周辺回路 13a、Stream Graphics Processor 14、Object Buffer 15、Composition Buffer 16、Graphical Controller 17 から構成される。

- 20      Coded Data Buffer 13 は、機能セグメントがDTS、PTSと共に格納されるバッファである。かかる機能セグメントは、Transport Buffer 4a,b,cに格納されたトランスポートストリームの各TSパケットから、TSパケットヘッダ、PESパケットヘッダを取り除き、ペイロードをシークエンシャルに配列することにより得られたものである。取り除かれたTSパケットヘッダ、PESパケットヘッダのうち、PTS/DTSは、PESパケットと対応付けて格納される。

- 25      周辺回路 13aは、Coded Data Buffer 13 - Stream Graphics Processor 14 間の転送、Coded Data Buffer 13 - Composition Buffer 16 間の転送を実現するワイヤロジックである。この転送処理において現在時点がODSのDTSに示される時刻になれば、ODSを、Coded Data Buffer 13 からStream Graphics Processor 14 に転送する。また現在時刻がPCS、PDSのDTSに示される時刻になれば、PCS、PDSをComposition Buffer 16 に転送するという処理を行う。

- 30      Stream Graphics Processor 14 は、ODSをデコードして、デコード

により得られたインデックスカラーからなる非圧縮状態の非圧縮グラフィックスをグラフィックスオブジェクトとしてObject Buffer 15に書き込む。Stream Graphicsプロセッサ 14によるデコードは瞬時に行われ、デコードによりグラフィックスオブジェクトをStream Graphics  
5 プロセッサ 14は一時的に保持する。Stream Graphicsプロセッサ 14によるデコードは瞬時になされるが、Stream Graphicsプロセッサ 14からObject Buffer 15への書き込みは、瞬時には終わらない。BD-ROM規格のプレーヤモデルでは、Object Buffer 15への書き込みは、128Mbpsという転送レートでなされるからである。Object Buffer  
10 15への書き込み完了時点は、ENDセグメントのPTSに示されているので、このENDセグメントのPTSに示される時点が経過するまで、次のDSに対する処理を待つことになる。各ODSをデコードすることにより得られたグラフィックスオブジェクトの書き込みは、そのODSに関連付けられたDTSの時刻に開始し、ODSに関連付けられたPTSに示されるデ  
15 コード終了時刻までに終了する。

書き込み時にあたってDS<sub>n</sub>側のグラフィックスデータと、DS<sub>n+1</sub>側のグラフィックスデータとに割り当てられているobject\_idが別々の場合、Stream Graphicsプロセッサ 14はDS<sub>n</sub>側のグラフィックスデータと、DS<sub>n+1</sub>側のグラフィックスデータとを、Object Buffer 15にお  
20 ける別々の領域に書き込む。これによりDS<sub>n</sub>のPCSにより参照されるグラフィックスオブジェクトは、DS<sub>n+1</sub>に属するグラフィックスオブジェクトにより上書きされることなく、パイプラインで表示に供される。両グラフィックスオブジェクトに割り当てられているobject\_idが同じである場合、前記Stream Graphicsプロセッサ 14は、Object Buffer 15  
25 において先行DS側のグラフィックスデータが格納されている領域と同じ領域に、後続DS側のグラフィックスデータを上書きする。かかる場合、パイプラインは行わない。またDSに属するグラフィックスオブジェクトには、同一DSのPCSにより参照されているものと、参照されていないものとがある。Stream Graphicsプロセッサ 14は、PCSにより  
30 参照されているグラフィックスオブジェクトだけでなく、参照されてな

いグラフィクスオブジェクトを逐次デコードして、デコードにより得られたグラフィクスをObject Buffer 15に格納する。

Object Buffer 15は、ETSI EN 300 743標準規格におけるピクセルバッファに相当するバッファであり、Stream Graphics Processor 14のデコードにより得られたグラフィクスオブジェクトが配置される。Object Buffer 15は、グラフィクスプレーン8の2倍／4倍の大きさに設定せねばならない。何故ならScrollingを実現する場合を考えると、グラフィクスプレーン8の2倍、4倍のグラフィクスオブジェクトを格納しておかねばならないからである。

10 Composition Buffer 16は、PCS、PDSが配置されるメモリである。一処理すべきDisplay Setが2つあり、これらの有効区間が重複している場合、Compositionバッファ16には処理すべきPCSが複数格納される。

15 Graphical Controller 17は、Graphicalコントローラ17はPCSの解釈を行い、PCSの解釈結果に従って、グラフィクスオブジェクトのObject Buffer 15への書き込み、及び、Object Buffer 15からのグラフィクスオブジェクトの読み出し、グラフィクスオブジェクトの表示を実行する。Graphicalコントローラ17による表示は、PCSを格納したPESパケットのPTSに示される時点において実行される。

20 オブジェクトCによる画面構成を実行するにあたって、Graphicsコントローラ17は以下の処理を行う。Graphicsコントローラ17は、オブジェクトCに記述されたグラフィクスオブジェクト識別子の参照値(object\_id\_ref)を読み取り、そのobject\_id\_refにより指示されるグラフィクスオブジェクトのうち、

25 object\_cropping\_horizontal\_position, object\_cropping\_vertical\_positionから、object\_cropping\_width、object\_cropping\_heightに示されるクロップ範囲を切り出して、グラフィックスプレーンのobject\_horizontal\_position, object\_vertical\_positionにより示される位置に書き込む。この参照値が、オブジェクトCと同じDisplay Set  
30 に属するグラフィクスオブジェクトを表している場合、同じDisplay

Setに属するODSがデコードされた後でないと、上述したグラフィックスプレーンへの書き込みは行えない。

一方、オブジェクトCが属するDisplay SetのComposition\_stateがNormal Caseである場合、デコードにより得られたグラフィックスオブジェクトが、前もってオブジェクトバッファ内に格納されているので、ODSのデコードを待つことなく、上述したグラフィックスプレーンへの書き込みを行うことができる。

PalletOnly Display Update時におけるGraphicsコントローラ17の読み出し処理は以下の通りである。Display Setが、BD-ROMから読み出された場合、Graphicsコントローラ17は、PCSのPalette\_update\_flagが、“1”であるか否かを判定する。もし1であるなら、PCSのpallet\_idに記述されているパレットデータを用いた画素値変換を、CLUT部9に命じた上で、Object Buffer15からグラフィックスオブジェクトを読み出し、グラフィックスプレーン8に格納させてゆく。CLUT部9に対するパレットデータの設定のみを行うことで、グラフィックスの更新を行うのである。

続いて、PIDフィルタ3、Transport Buffer4a,b,c、グラフィックスプレーン8、CLUT部9、Coded Data Buffer13～Graphical Controller17を構成するための、転送レート、バッファサイズの推奨値について説明する。図29は、書込レートRx,Rc,Rd、グラフィックスプレーン8、Coded Data Buffer13、Object Buffer15、Composition Buffer16のサイズを示す図である。

Object Buffer15－グラフィックスプレーン8間の転送レートRcは、本装置において最も高い転送レートであり、ウィンドウサイズ、フレームレートから256Mbps(=500Kバイト×29.97×2)と算出される。

Stream Graphics Processor14－Object Buffer15間の転送レートRd(Pixel Decoding Rate)は、Rcとは異なり、ビデオフレームの周期によるアップデートは要求されずRcの1/2,1/4でよい。故に128Mbps,64Mbpsになる。

Transport Buffer4a,b,c－Coded Data Buffer13間のTransport



Buffer LeakレートRxは、圧縮状態たるODSの転送レートである。従ってTransport Buffer Leakレートは、Rdに圧縮率を乗じた転送レートでよい。ODSの圧縮率を25%と仮定すれば、16Mbps(=64Mbps×25%)で足りる。

- 5     この図に示す転送レート、バッファ規模はあくまでもミニマムスタンダードであり、これより大きい値での実装を否定している訳ではない。

以上のように構成された再生装置において、各構成要素はパイプライン式にデコード処理を行うことができる。

- 10     図30は、再生装置によるパイプライン処理を示すタイミングチャートである。第5段目は、BD-ROMにおけるDisplay Setを示し、第4段目は、Coded Data Buffer 13へのPCS、WDS、PDS、ODSの読出期間を示す。第3段目は、Stream Graphics Processor 14による各ODSのデコード期間を、第2段目はComposition Buffer 16の格納内容を、  
15     第1段目はGraphical Controller 17の処理内容を示す。

- ODS1,2に付与されたDTS(デコード開始時刻)は、図中のt31, t32の時点を示している。デコード開始時刻がDTSに規定されているので、各ODSは、自身のDTSに示される時刻までにCoded Data Buffer 13に読み出されなければならない。そのためODS1の読み出しは、Coded Data  
20     Buffer 13へのODS1のデコード期間dp1の直前までに完了している。Coded Data Buffer 13へのODS2の読み出しは、ODS2のデコード期間dp2の直前までに完了している。

- 一方、ODS1,2に付与されたPTS(デコード終了時刻)は、図中のt32, t33の時点を示している。Stream Graphics Processor 14による  
25     ODS1のデコードはt32までに完了し、ODS2のデコードは、t33に示される時刻までに完了する。以上のように、Stream Graphics Processor 14は、各ODSのDTSに示される時刻までに、ODSをCoded Data Buffer 13に読み出し、Coded Data Buffer 13に読み出されたODSを、各ODSのPTSに示される時刻までに、デコードしてObject Buffer 15に書き  
30     込む。

本図の第1段目における期間cd1は、Graphics Controller17がグラフィックスプレーン8をクリアするのに要する期間である。また期間td1は、Object Buffer15上にえられたグラフィックスオブジェクトを、グラフィックスプレーン8に書き込むのに要する期間である。WDSのPTSは、この書き込みの開始にあたってのデッドラインを示し、PCSのPTSはこの書き込みの終了時点及び表示タイミングを示す。このPCSのPTSに示される時点になれば、対話画面を構成する非圧縮グラフィックスがグラフィックスプレーン8上に得られることになる。この非圧縮グラフィックスの色変換をCLUT部9に行わせ、ビデオプレーン6に格納されている非圧縮ピクチャとの合成を加算器10に行わせれば、合成画像が得られることになる。

グラフィックスデコーダ12において、Graphics Controller17がグラフィックスプレーン8のクリアを実行している間においても、Stream Graphics Processor14のデコードは継続して行われる。以上のようなパイプライン処理により、グラフィックスの表示を迅速に実施することができる。

図30では、グラフィックスプレーンのクリアが、ODSのデコードより早く終わる場合を想定したが、図31は、ODSのデコードが、グラフィックスプレーンのクリアより早く終わる場合を想定したパイプライン処理を示すタイミングチャートである。この場合、ODSのデコードが完了した段階では、グラフィックスプレーンへの書き込みを実行することができず、グラフィックスプレーンのクリアが完了した時点で、デコードにより得られたグラフィックスをグラフィックスプレーンに書き込むことができる。

再生装置におけるバッファ占有量の時間的遷移について図32を参照しながら説明する。図32は、図28におけるCompositionバッファ16、Object Buffer15、Coded Dataバッファ13、グラフィックスプレーン8の時間的遷移を示すタイミングチャートである。本図は、第1段目から第4段目までに、グラフィックスプレーン8、Object Buffer15、Coded Dataバッファ13、Compositionバッファ16に

における占有量の時間的遷移を示している。この占有量の時間的遷移は、横軸を時間軸とし、縦軸を占有量とした折れ線グラフの表記で表現している。

図 3 2 の第 4 段目は、Compositionバッファ 1 6 における占有量の時間的遷移を示す。本図に示すようにCompositionバッファ 1 6 の時間的遷移は、Coded Dataバッファ 1 3 から出力されPCSが格納されることによる単調増加vf0を含む。

第 3 段目は、Coded Dataバッファ 1 3 における占有量の時間的遷移を示す。本図に示すようにCoded Dataバッファ 1 3 の時間的遷移は、ODSが格納されることによる単調増加Vf1,Vf2と、格納されたODSが順次Stream Graphicsプロセッサ 1 4 により取り出されることによる単調減少Vg1,Vg2とを含む。単調増加Vf1,Vf2の傾きは、Transportバッファ 4 a,b,cからCoded Dataバッファ 1 3 への出力レートRxに基づき、単調減少Vg1,Vg2の傾きは、Stream Graphicsプロセッサ 1 4 によるデコードであり、瞬時に実行される。つまりODSに対するデコードは瞬時に行われ、Stream Graphicsプロセッサ 1 4 は、デコードにより得られた非圧縮グラフィクスを保持する。Stream Graphicsプロセッサ 1 4 からObject Buffer 1 5 への伝送路の書込レートは128Mbpsであるため、この書込レートにより、Object Buffer 1 5 の占有量は増加する。

第 2 段目は、Object Buffer 1 5 における占有量の時間的遷移を示す。本図に示すようにObject Buffer 1 5 の時間的遷移は、Stream Graphicsプロセッサ 1 4 から出力されたODSが格納されることによる単調増加Vh1,Vh2を含む。単調増加Vh1,Vh2の傾きは、Stream Graphicsプロセッサ 1 4 からObject Buffer 1 5 への転送レートRdに基づく。第 3 段目の単調減少が生じる期間及びの第 2 段目の単調増加が生ずる期間が、デコード期間である。このデコード期間の始期は、ODSのDTSに示されており、デコード期間の終期は、ODSのPTSに示されている。このODSのPTSに示される期間までに、非圧縮のグラフィクスがobject buffer 1 5 に格納されれば、ODSに対するデコードは完了した

ことになる。ODSのPTSに示される期間までに、非圧縮のグラフィクスがobject buffer 15に格納されることが、必須の要件であり、このデコード期間における単調増加、単調減少はどのようなものであってもよい。

- 5 第1段目は、グラフィクスプレーン8における占有量の時間的遷移を示す。本図に示すようにグラフィクスプレーン8の時間的遷移は、Object Buffer 15から出力されたデコード済みODSが格納されることによる単調増加Vf3を含む。単調増加Vf3の傾きは、Object Buffer 15からグラフィクスプレーン8への転送レートRcに基づく。この単調増加の終期は、PCSのPTSに示されている。

- 10 ODSに付与されたDTS、PTS、ICSに付与されたDTS、PTS、そして図29に示した各バッファのサイズ、転送レートを用いれば、本図のようなグラフを作図することにより、BD-ROMにて供給すべきAVClipの再生時において、各バッファの状態がどのように変化するかが、オーサリングの段階で明らかになる。

- 15 このバッファ状態の遷移は、DTS、PTSに示される値を書き換えることで、調整することが可能なので、再生装置側のデコードのスペックを越えるような復号負荷の発生を回避したり、再生にあたってのバッファオーバーフローの回避することができる。そのため再生装置の開発にあたってのハードウェア、ソフトウェアの実装が簡易になる。

- 20 以上が再生装置の内部構成である。続いてグラフィクスデコーダ12を、どのようにして実装するかについて説明する。グラフィクスデコーダ12は、図33の処理手順を行うプログラムを作成し、CPUに実行させることにより実装可能である。以降、図33を参照しながら、制御部20の処理手順について説明する。

- 25 図33は、機能セグメントのロード処理の処理手順を示すフローチャートである。本フローチャートにおいてSegmentKとは、AVClipの再生時において、読み出されたSegment(PCS, WDS, PDS, ODS)のそれぞれを意味する変数であり、無視フラグは、このSegmentKを無視するかロードするかを切り換えるフラグである。本フローチャートは、無視フラ

グを0に初期化した上で、ステップS 2 1～S 2 4、ステップS 2 7～S 3 1の処理を全てのSegmentKについて繰り返すループ構造を有している(ステップS 2 5、ステップS 2 6)。

5 ステップS 2 1は、SegmentKがPCSであるか否かの判定であり、もしSegmentKがPCSであれば、ステップS 2 7、ステップS 2 8の判定を行う。

10 ステップS 2 2は、無視フラグが1かどうかの判定である。無視フラグが0であるならステップS 2 3に移行し、1であるならステップS 2 4に移行する。無視フラグが1であれば(ステップS 2 2でNo)、ステップS 2 3においてSegmentKをCoded Data Buffer 1 3にロードする。

15 無視フラグが0に設定されていれば(ステップS 2 2がNo)、ステップS 2 4においてSegmentKが無視される。これによりDSに属する機能セグメントは全て、ステップS 2 2がNoになって、無視されることになる(ステップS 2 4)。

このように、SegmentKが無視されるか、ロードされるかは、無視フラグの設定により決まる。ステップS 2 7～S 3 1、S 3 4、S 3 5は、この無視フラグを設定する処理である。

20 ステップS 2 7は、PCSにおけるcomposition\_stateがAcquisition Pointであるか否かの判定である。SegmentKがAcquisition Pointであるなら、ステップS 2 8に移行し、SegmentKがもしEpoch StartかNormal Caseであるなら、ステップS 3 1に移行する。

25 ステップS 2 8は、先行するDSがグラフィクスデコーダ1 2内のどれかのバッファ(Coded Data Buffer 1 3、Stream Graphicsプロセッサ1 4、Object Buffer 1 5、Composition Buffer 1 6)に存在するかどうかの判定であり、ステップS 2 7がYesである場合に実行される。グラフィクスデコーダ1 2内にDSが存在しないケースとは、頭出しがなされたケースをいう。この場合、Acquisition PointたるDSから、表示を開始せねばならないので、ステップS 3 0に移行する(ステップS 2 8でNo)。

30

ステップS 3 0は、無視フラグを0に設定し、ステップS 2 2に移行する。

グラフィクスデコーダ1 2内にDSが存在するケースとは、通常再生がなされたケースをいう。この場合、ステップS 2 9に移行する(ステップS 2 8でYes)。ステップS 2 9は、無視フラグを1に設定し、

5    ステップS 2 2に移行する。

ステップS 3 1は、PCSにおけるcomposition\_stateがNormal Caseであるか否かの判定である。もしNormal Caseであるなら、ステップS 3 4に移行する。SegmentKがEpoch Startであるなら、ステップS

10    3 0において無視フラグを0に設定する。    ステップS 3 4は、ステップS 2 8と同じであり、先行するDSがグラフィクスデコーダ1 2内に存在するかどうかの判定を行う。もし存在するなら、無視フラグを0に設定する(ステップS 3 0)。存在しないなら、元々、対話画面を構成する充分な機能セグメントが得られないため、無視フラグを1に

15    設定する(ステップS 3 5)。かかるフラグ設定により、先行するDSがグラフィクスデコーダ1 2に存在しない場合、Normal Caseを構成する機能セグメントは無視されることになる。

DSが、図3 4のように多重化されている場合を想定して、DSの読み出しがどのように行われるかを説明する。図3 4の一例では、3つの

20    DSが動画と多重化されている。この3つのDSのうち、初めのDS1は、composition\_stateがEpoch\_Start、DS10はAcquisition Point、DS20は、Normal Caseである。

かかる3つのDSが、動画と多重化されているAVClipにおいて、ピクチャデータpt10からの頭出しが矢印amlに示すように行われたものと

25    する。この場合、頭出し位置に最も近いDS10が、図3 3のフローチャートの対象となる。ステップS 2 7においてcomposition\_stateはAcquisition Pointと判定されるが、先行するDSはCoded Data Buffer 1 3上に存在しないため、無視フラグは0に設定され、このDS10が図3 5の矢印mdlに示すように再生装置のCoded Data Buffer 1 3にロー

30    ドされる。一方、頭出し位置がDS10の存在位置より後である場合(図

3 4 の矢印am2)、DS20は、Normal CaseのDisplay Setであり、先行するDS20はCoded Data Buffer 1 3 に存在しないので、このDisplay Setは、無視されることになる(図 3 5 の矢印md2)。

図 3 6 のように通常再生が行われた場合のDS1, 10, 20のロードは、  
5 図 3 6 に示すものとなる。3つのDSのうち、PCSのcomposition\_stateがEpoch StartであるDS1は、そのままCoded Data Buffer 1 3 にロードされるが(ステップ S 2 3)、PCSのcomposition\_stateがAcquisition PointであるDS10については、無視フラグが1に設定されるため(ステップ S 2 9)、これを構成する機能セグメントはCoded  
10 Data Buffer 1 3 にロードされず無視される(図 3 7 の矢印rd2, ステップ S 2 4)。またDS20については、PCSのcomposition\_stateはNormal Caseなので、Coded Data Buffer 1 3 にロードされる(図 3 7 の矢印rd3)。

続いてGraphical Controller 1 7 の処理手順について説明する。図  
15 3 8 ~ 図 4 0 は、Graphical Controller 1 7 の処理手順を示すフローチャートである。

ステップ S 4 1 ~ ステップ S 4 4 は、本フローチャートのメインルーチンであり、ステップ S 4 1 ~ ステップ S 4 4 に規定した何れかの事象の成立を待つ。

20 ステップ S 4 1 は、現在の再生時点がPCSのDTS時刻になっているか否かの判定であり、もしなっていれば、ステップ S 4 5 ~ ステップ S 5 3 の処理を行う。

ステップ S 4 5 は、PCSのcomposition\_stateが、Epoch\_Startであるか否かの判定であり、もしEpoch\_Startであるなら、ステップ S 4  
25 6 においてグラフィクスプレーン 8 を全クリアする。それ以外であるなら、ステップ S 4 7 においてWDSのwindow\_horizontal\_position、window\_vertival\_position、window\_width、window\_heightに示されるwindowをクリアする。

ステップ S 4 8 は、ステップ S 4 6 又はステップ S 4 7 でのクリア  
30 後の実行されるステップであり、任意のODSxのPTS時刻が既に経過し

ているか否かの判定である。つまり、グラフィクスプレーン 8 全体のクリアにあたっては、そのクリア時間に長時間を費するので、ある ODS(ODSx)のデコードが既に完了していることもある。ステップ S 4 8 はその可能性を検証している。もし経過していないなら、メインルーチンにリターンする。どれかの ODS のデコード時刻を経過しているなら、ステップ S 4 9 ～ステップ S 5 1 を実行する。ステップ S 4 9 は、object\_cropped\_flag が 0 を示しているか否かの判定であり、もし 0 を示しているなら、グラフィクスオブジェクトを非表示とする(ステップ S 5 0)。

- 10      もし 0 を示していないなら、object\_cropping\_horizontal\_position、object\_cropping\_vertival\_position、cropping\_width、cropping\_height に基づきクロップされたグラフィクスオブジェクトを、グラフィクスプレーン 8 の window において object\_horizontal\_position、object\_vertival\_position に示される位置に書き込む(ステップ S 5 1)。以上の処理により、ウィンドウに 1 つ以上のグラフィクスオブジェクトが描かれることになる。

- 20      ステップ S 5 2 は、別の ODSy の PTS 時刻が経過しているか否かの判定である。ODSx をグラフィクスプレーン 8 に書き込んでいる際、別の ODS のデコードが既に完了していれば、この ODSy を ODSx にして(ステップ S 5 3)、ステップ S 4 9 に移行する。これにより、別の ODS に対しても、ステップ S 4 9 ～S 5 1 の処理が繰り返し行われる。

次に図 3 9 を参照して、ステップ S 4 2、ステップ S 5 4 ～ステップ S 5 9 について説明する。

- 25      ステップ S 4 2 は、現在の再生時点が WDS の PTS であるか否かの判定であり、もし WDS の PTS であるなら、ステップ S 5 4 においてウィンドウが 1 つであるか否かを判定し、もし 2 つであれば、メインルーチンにリターンする。ウィンドウが 1 つであるなら、ステップ S 5 5 ～ステップ S 5 9 のループ処理を行う。このループ処理は、ウィンドウに表示される 2 つのグラフィクスオブジェクトのそれぞれについて、ステップ S 5 7 ～ステップ S 5 9 を実行するというものである。ステップ
- 30



S 5 7 は、object\_cropped\_flagが0を示しているか否かの判定であり、もし0を示しているなら、グラフィクスオブジェクトを非表示とする(ステップ S 5 8)。

もし0を示していないなら、object\_cropping\_horizontal\_position、  
5 object\_cropping\_vertival\_position、cropping\_width、cropping\_heightに基づきクロップされたグラフィクスオブジェクトを、グラフィクスプレーン 8 のwindowにおいてobject\_horizontal\_position、object\_vertival\_positionに示される位置に書き込む(ステップ S 5 9)。以上の処理を繰り返せば、ウィンドウに1つ以上のグラフィクスオブジェクトが描かれることになる。

ステップ S 4 4 は、現在の再生時点がPCSのPTSに示される時点であるかの判定であり、もしそうであるなら、ステップ S 6 0 においてPallet\_update\_flagが1を示しているか否かを判定する。もし1を示しているなら、pallet\_idに示されるPDSをCLUT部に設定する(ステップ  
15 S 6 1)。0を示しているなら、ステップ S 6 1 をスキップする。

その後、グラフィクスプレーン 8 におけるグラフィクスオブジェクトの色変換をCLUT部に行わせて、動画像と合成する(ステップ S 6 2)。

次に図 4 0 を参照して、ステップ S 4 3、ステップ S 6 4 ~ ステップ S 6 6 について説明する。

20 ステップ S 4 3 は、現在の再生時点がODSのPTSであるか否かの判定であり、もしODSのPTSであるなら、ステップ S 6 3 においてウィンドウが2つであるか否かを判定する。もし1つであれば、メインルーチンにリターンする。

ステップ S 4 3 及びステップ S 6 3 の判定は以下の意味をもつ。つまりウィンドウが2つある場合、それぞれのウィンドウには、1つずつ  
25 グラフィクスオブジェクトが表示される。そうすると、それぞれのODSのデコードが完了する度に、デコードにより得られたグラフィクスオブジェクトを、グラフィックスプレーンに書き込んでゆく処理が必要になる(例:図 1 9 (b) のケース)。そこで現在時点がODSのPTSに示  
30 される時点であり、ウィンドウが2つであるなら、個々のグラフィク

5 スオブジェクトをグラフィックスプレーンに書き込むべく、ステップ  
S 6 4～ステップ S 6 6 を実行する。ステップ S 6 4 は、  
object\_cropped\_flagが0を示しているか否かの判定であり、もし示し  
ているなら、グラフィックスオブジェクトを非表示とする(ステップ S  
6 5)。

もし0を示していないなら、object\_cropping\_horizontal\_position、  
object\_cropping\_vertival\_position、cropping\_width、  
cropping\_heightに基づきクロップされたグラフィックスオブジェクト  
を、グラフィックスプレーン 8 のwindowにおいて  
10 object\_horizontal\_position, object\_vertival\_positionに示される  
位置に書き込む(ステップ S 6 6)。以上の処理を繰り返せば、各ウィ  
ンドウにグラフィックスオブジェクトが描かれることになる。

#### (第2実施形態)

本実施形態は、BD-ROMの製造工程に関する実施形態である。図 4 1  
15 は、第1実施形態に示したPCSを作成するための製造工程を示す図で  
ある。

BD-ROMの制作工程は、動画収録、音声収録等の素材作成を行う素材  
制作工程 S 2 0 1、オーサリング装置を用いて、アプリケーションフ  
ォーマットを生成するオーサリング工程 S 2 0 2、BD-ROMの原盤を作  
20 成し、プレス・貼り合わせを行って、BD-ROMを完成させるプレス工程  
S 2 0 3を含む。

これらの工程のうち、BD-ROMを対象としたオーサリング工程は、以  
下のステップ S 2 0 4～ステップ S 2 1 0を含む。

ステップ S 2 0 4において制御情報、ウィンドウ定義情報、パレッ  
25 ト定義情報、グラフィックスを記述し、ステップ S 2 0 5では、制御情  
報、ウィンドウ定義情報、パレット定義情報、グラフィックスを機能セグ  
メントに変換する。そしてステップ S 2 0 6において同期したいピク  
チャが出現するタイミングに基づき、PCSのPTSを設定し、ステップ S  
2 0 7では、PTS[PCS]の値に基づき、DTS[ODS], PTS[ODS]を設定する。  
30 ステップ S 2 0 8において、DTS[ODS]の値に基づき、

DTS[PCS], PTS[PDS], DTS[WDS], PTS[WDS]を設定し、ステップS209では、プレーヤモデルにおける各バッファの占有量の時間的遷移をグラフ化する。ステップS210では、グラフ化された時間的遷移がプレーヤモデルの制約を満たすか否かを判定し、もし満たさないなら、

5 ステップS211において各機能セグメントのDTS、PTSを書き換える。もし満たすならステップS212においてグラフィックスストリームを生成し、ステップS213においてグラフィックスストリームを別途生成されたビデオストリーム、オーディオストリームと多重してAVClipを得る。以降、AVClipをBD-ROMのフォーマットに適合させることにより、アプリケーションフォーマットが完成する。

10

(備考)

以上の説明は、本発明の全ての実施行為の形態を示している訳ではない。下記(A)(B)(C)(D)……の変更を施した実施行為の形態によっても、本発明の実施は可能となる。本願の請求項に係る各発明は、以上に記載した複数の実施形態及びそれらの変形形態を拡張した記載、

15 ないし、一般化した記載としている。拡張ないし一般化の程度は、本発明の技術分野の、出願当時の技術水準の特性に基づく。

(A)全ての実施形態では、本発明に係る記録媒体をBD-ROMとして実施したが、本発明の記録媒体は、記録されるグラフィックスストリームに特徴があり、この特徴は、BD-ROMの物理的性質に依存するものではない。グラフィックスストリームを記録しうる記録媒体なら、どのような記録媒体であってもよい。例えば、

20 DVD-ROM, DVD-RAM, DVD-RW, DVD-R, DVD+RW, DVD+R, CD-R, CD-RW等の光ディスク、PD, MO等の光磁気ディスクであってもよい。また、コンパクトフラッシュカード、スマートメディア、メモリスティック、マルチメディアカード、PCM-CIAカード等の半導体メモリカードであってもよい。フレキシブルディスク、SuperDisk, Zip, Clik!等の磁気記録ディスク(i)、ORB, Jaz, SparQ, SyJet, EZFley, マイクロドライブ等のリムーバブルハードディスクドライブ(ii)であってもよい。更に、機器内蔵

25

30 型のハードディスクであってもよい。

(B)全ての実施形態における再生装置は、BD-ROMに記録されたAVClipをデコードした上でTVに出力していたが、再生装置をBD-ROMドライブのみとし、これ以外の構成要素をTVに具備させてもよい、この場合、再生装置と、TVとをIEEE1394で接続されたホームネットワークに組み入れることができる。また、実施形態における再生装置は、テレビと接続して利用されるタイプであったが、ディスプレイと一体型となった再生装置であってもよい。更に、各実施形態の再生装置において、処理の本質的部分をなすシステムLSI(集積回路)のみを、実施としてもよい。これらの再生装置及び集積回路は、何れも本願明細書に記載された発明であるから、これらの何れの態様であろうとも、第1実施形態に示した再生装置の内部構成を元に、再生装置を製造する行為は、本願の明細書に記載された発明の実施行為になる。第1実施形態に示した再生装置の有償・無償による譲渡(有償の場合は販売、無償の場合は贈与になる)、貸与、輸入する行為も、本発明の実施行為である。店頭展示、カタログ勧誘、パンフレット配布により、これらの譲渡や貸渡を、一般ユーザに申し出る行為も本再生装置の実施行為である。

(C)各フローチャートに示したプログラムによる情報処理は、ハードウェア資源を用いて具体的に実現されていることから、上記フローチャートに処理手順を示したプログラムは、単体で発明として成立する。全ての実施形態は、再生装置に組み込まれた態様で、本発明に係るプログラムの実施行為についての実施形態を示したが、再生装置から分離して、第1実施形態に示したプログラム単体を実施してもよい。プログラム単体の実施行為には、これらのプログラムを生産する行為(1)や、有償・無償によりプログラムを譲渡する行為(2)、貸与する行為(3)、輸入する行為(4)、双方向の電子通信回線を介して公衆に提供する行為(5)、店頭、カタログ勧誘、パンフレット配布により、プログラムの譲渡や貸渡を、一般ユーザに申し出る行為(6)がある。

(D)各フローチャートにおいて時系列に実行される各ステップの「時」の要素を、発明を特定するための必須の事項と考える。そうする

と、これらのフローチャートによる処理手順は、再生方法の使用形態を開示していることがわかる。各ステップの処理を、時系列に行うことで、本発明の本来の目的を達成し、作用及び効果を奏するよう、これらのフローチャートの処理を行うのであれば、本発明に係る記録方法の実施行為に該当することはいうまでもない。

(E)BD-ROMに記録するにあたって、AVClipを構成する各TSパケットには、拡張ヘッダを付与しておくことが望ましい。拡張ヘッダは、TP\_extra\_headerと呼ばれ、『Arribval\_Time\_Stamp』と、

『copy\_permission\_indicator』とを含み4バイトのデータ長を有する。

- 10 TP\_extra\_header付きTSパケット(以下EX付きTSパケットと略す)は、32個毎にグループ化されて、3つのセクタに書き込まれる。32個のEX付きTSパケットからなるグループは、6144バイト( $=32 \times 192$ )であり、これは3個のセクタサイズ6144バイト( $=2048 \times 3$ )と一致する。3個のセクタに収められた32個のEX付きTSパケットを"Aligned Unit"という。
- 15 IEEE1394を介して接続されたホームネットワークでの利用時において、再生装置は、以下のような送信処理にてAligned Unitの送信を行う。つまり送り手側の機器は、Aligned Unitに含まれる32個のEX付きTSパケットのそれぞれからTP\_extra\_headerを取り外し、TSパケット本体をDTCP規格に基づき暗号化して出力する。TSパケットの出力に
- 20 あたっては、TSパケット間の随所に、isochronousパケットを挿入する。この挿入箇所は、TP\_extra\_headerのArribval\_Time\_Stampに示される時刻に基づいた位置である。TSパケットの出力に伴い、再生装置はDTCP\_Descriptorを出力する。DTCP\_Descriptorは、TP\_extra\_headerにおけるコピー許否設定を示す。ここで「コピー禁止」を示すよう
- 25 DTCP\_Descriptorを記述しておけば、IEEE1394を介して接続されたホームネットワークでの利用時においてTSパケットは、他の機器に記録されることはない。

- (F)各実施形態におけるデジタルストリームは、BD-ROM規格の
- 30 AVClipであったが、DVD-Video規格、DVD-Video Recording規格の

VOB(Video Object)であってもよい。VOBは、ビデオストリーム、オーディオストリームを多重化することにより得られたISO/IEC13818-1規格準拠のプログラムストリームである。またAVClipにおけるビデオストリームは、MPEG4やWMV方式であってもよい。更にオーディオストリームは、Linear-PCM方式、Dolby-AC3方式、MP3方式、MPEG-AAC方式、dts方式であってもよい。

(G)各実施形態における映画作品は、アナログ放送で放送されたアナログ映像信号をエンコードすることにより得られたものでもよい。デジタル放送で放送されたトランスポートストリームから構成されるストリームデータであってもよい。

またビデオテープに記録されているアナログ／デジタルの映像信号をエンコードしてコンテンツを得ても良い。更にビデオカメラから直接取り込んだアナログ／デジタルの映像信号をエンコードしてコンテンツを得ても良い。他にも、配信サーバにより配信されるデジタル著作物でもよい。

(H)第1実施形態～第2実施形態に示したグラフィックスオブジェクトは、ランレングス符号化されたラスタデータである。グラフィックスオブジェクトの圧縮・符号化方式にランレングス符号方式を採用したのは、ランレングス符号化は字幕の圧縮・伸長に最も適しているためである。字幕には、同じ画素値の水平方向の連続長が比較的長くなるという特性があり、ランレングス符号化による圧縮を行えば、高い圧縮率を得ることができる。また伸長のための負荷も軽く、復号処理のソフトウェア化に向いているからである。しかし、グラフィックスオブジェクトにランレングス符号化方式を採用したというのは、本発明の必須事項ではなく、グラフィックスオブジェクトはPNGデータであってもよい。またラスタデータではなくベクタデータであってもよい、更に透明な絵柄であってもよい。

(I)PCSによる表示効果の対象は、装置側の言語設定に応じて選ばれた字幕のグラフィックスであってもよい。これにより、現状のDVDにおいて動画像本体で表現していたような文字を用いた表示効果を、装置

側の言語設定に応じて表示された字幕グラフィックスで実現することができるので、実用上の価値は大きい。

またPCSによる表示効果の対象は、装置側のディスプレイ設定に応じて選ばれた字幕グラフィックスであってもよい。つまり、ワイドビジョン、パンスキャン、レターボックス用といった様々な表示モード用のグラフィックスがBD-ROMに記録されており、装置側は自身に接続されたテレビの設定に応じてこれらの何れかを選んで表示する。この場合、そうして表示された字幕グラフィックスに対し、PCSに基づく表示効果をほどこすので、見栄えがよくなる。これにより、動画像本体で表現していたような文字を用いた表示効果を、装置側のディスプレイ設定に応じて表示された字幕で実現することができるので、実用上の価値は大きい。

(J)第1実施形態ではグラフィックスプレーンへの書込レート $R_c$ は、1ビデオフレーム内にグラフィックスプレーンクリア及び再描画が可能になるよう、windowのサイズを全体の25%に定めたが、これらクリア・再描画が垂直帰線期間に完遂するよう、 $R_c$ を定めても良い。垂直帰線期間は1/29.93秒の25%と仮定すると、 $R_c$ は1Gbpsになる。 $R_c$ をこのように設定することでグラフィックス表示はスムーズになされるので、実上の効果は大きい。

また垂直帰線期間での書き込みに加え、ラインスキャンに同期した書き込みを併用してもよい。これにより、 $R_c=256\text{Mbps}$ の書込レートであっても、スムーズな表示の実現が可能になる。

(K)各実施形態において再生装置には、グラフィックスプレーンを実装したが、このグラフィックスプレーンに代えて、一ライン分の非圧縮画素を格納するラインバッファを具備してもよい。映像信号への変換は水平行(ライン)毎に行われるので、このラインバッファさえ具備していれば、この映像信号への変換は行なえるからである。

(L)グラフィックスたる字幕は、映画の台詞を表す文字列であるとして説明を進めたが、動画と緻密に同期して表示されるものであればなんであってもよい。例えばイラストや絵柄、キャラクタ、シンボルマ

ークを描いたものであってもよい。商標を構成するような図形、文字、色彩の組合せや、国の紋章、旗章、記章、国家が採用する監督/証明用の公の記号・印章、政府間国際機関の紋章、旗章、記章、特定商品の原産地表示を含んでいてもよい。

5 (M)第1実施形態では、字幕を画面の上側、下側に横書きで表示するものとして、ウィンドウをグラフィックスプレーンの上側、下側に定義したが、字幕を画面の右側、左側に表示するものとして、ウィンドウをグラフィックスプレーンの右側、左側に定義してもよい。こうすることにより、日本語字幕を縦書きで表示することができる。

10 (O)グラフィクスデコーダ12が、 $DS_n$ 及び $DS_{n+1}$ に対する処理をパイプライン式に行うのは、 $DS_n$ 及び $DS_{n+1}$ が、グラフィックスストリームにおける同じEpochに帰属している場合であり、前記 $DS_n$ 及び $DS_{n+1}$ が、互いに異なるEpochに属している場合、 $DS_n$ におけるグラフィクス表示を開始した後に、 $DS_{n+1}$ に対する処理を開始する。

15 またグラフィックスストリームには、動画との同期を主たる目的としたプレゼンテーション系のものと、対話的な表示を主目的としたインタラクティブ系のものとがあり、前記グラフィクスデコーダは、グラフィックスストリームがプレゼンテーション系である場合に、2つのDSに対するパイプライン式を行い、グラフィックスストリームがインタラクティブ系である場合、2つのDSに対するパイプライン式を行わない。

(P)Display Set内にグラフィクスデータを配列させるにあたって、図20では、同じDisplay Set内のPCSにより参照されるグラフィクスデータを、参照されないグラフィクスデータの前に配列したが、複数の参照グラフィクスデータ、及び、複数の非参照グラフィクスデータを、それぞれ、object\_idにより指示される識別番号順に配列してもよい。また複数の参照グラフィクスデータは、スキャンライン順に配列してもよい。スキャンラインとは、ディスプレイでの表示時において、走査される順番を表す。通常ディスプレイにおける走査は、左上→右下方向に向けてなされるから、表示座標が左上にあるグラフィクスデータを前に、表示座標が右下にあるグラフィクスデータを後に配

25

30



置するのである。こうすることで、グラフィクスデータ表示の迅速化を図ることができる。

- 5 上述した変更実施は可能であるものの、請求項に係る各発明は、従来技術の技術的課題を解決するための手段を反映したものであるから、請求項に係る各発明の技術範囲は、従来技術の技術的課題解決が当業者により認識される技術範囲を超えることはない。故に、本願の請求項に係る各発明は、詳細説明の記載と、実質的な対応関係を有する。

#### 産業上の利用可能性

- 10 本発明に係る記録媒体及び再生装置は、上記実施形態に内部構成が開示されており、この内部構成に基づき工業的に量産することが可能なので、資質において工業上用することができる。このことから本発明に係る記録媒体及び再生装置は、産業上利用可能性を有する。

## 請求の範囲

1. 動画ストリームとグラフィクスストリームとを多重化することにより得られたデジタルストリームが記録されている記録媒体であって、

- 5      グラフィクスストリームは、パケットの配列であり、  
パケットには、グラフィクスデータを格納したものと、制御情報を格納したものがあり、

10      制御情報は、パケット列において自身より前方に存在するグラフィクスデータを、所定のタイミングで動画ストリームと合成して表示する旨を示す、ことを特徴とする記録媒体。

2. グラフィクスストリームにおけるパケットは、複数のディスプレイセットの何れかに分類されており、個々のディスプレイセットは、グラフィクス表示を実現する一個のデータの集合であり、

- 15      前記グラフィクスデータ及び制御情報は、  
別々のディスプレイセットに属しており、  
前記グラフィクスデータは、同じディスプレイセットに属する制御情報からは参照されていない、非参照グラフィクスデータであることを特徴とする請求項1記載の記録媒体。

- 20      3. 制御情報は、完結した状態でパケットに格納されており、  
前記所望のタイミングは、前記パケットのタイムスタンプに示されている、ことを特徴とする請求項1記載の記録媒体。

4. 前記制御情報は、パレットデータと組みになって記録媒体に記録されており、

- 25      前記制御情報による表示は、パレットデータを用いたグラフィクスデータの色変換を、再生装置に命じる旨を示す、ことを特徴とする請求項1記載の記録媒体。

5. 制御情報により示される表示は、グラフィクスデータの2回目以降の表示であり、

- 30      制御情報は、更新フラグを有し、  
更新フラグは、色変換に用いるパレットデータのみを用いて表示更

新を行う旨を示す、ことを特徴とする請求項 4 記載の記録媒体。

6. 前記グラフィクスストリームはウィンドウ情報を含み、

ウィンドウ情報は、グラフィクス表示のための描画領域を特定する  
情報であり、動画ストリームと合成する際の画面上でのウィンドウの

5 位置、縦幅、横幅を示し、

前記制御情報による表示位置は、

前記グラフィクス表示がウィンドウに収まるように規定されてい  
る

ことを特徴とする請求項 1 記載の記録媒体。

10 7. ビデオストリーム、グラフィクスストリームが多重化されたデ  
ジタルストリームを再生する再生装置であって、

ビデオストリームをデコードして動画像を得るビデオデコーダと、  
グラフィクスストリームをデコードしてグラフィクスを得て、動画  
像に合成するグラフィクスデコーダとを備え、

15 グラフィクスデコーダは、

記録媒体から新たな制御情報が記録媒体から読み込まれれば、その  
制御情報に基づき、前もって送り込まれたグラフィクスデータを表示  
する

ことを特徴とする再生装置。

20 8. グラフィクスストリームはグラフィクス表示を実現するデータ  
の集合であるディスプレイセットを複数含み、

コントローラは、

1つのディスプレイセットが記録媒体から読み出されれば、そのデ  
ィスプレイセットに属するグラフィクスデータであって、同じデイス  
25 プレイセットに属する制御情報からは参照されていない、非参照グラ  
フィクスデータをオブジェクトバッファに格納しておき、

前記前もって送り込まれたグラフィクスデータは、オブジェクトバ  
ッファに格納されている非参照グラフィクスデータである

ことを特徴とする請求項 7 記載の再生装置。

30 9. 記録媒体から新たに読み出される制御情報は、完結した状態で

パケットに格納されており、

前記グラフィクスデコードによる表示は、

ビデオストリームの再生が、パケットのタイムスタンプに示されている時点に到達した際に行われる

5      ことを特徴とする請求項 7 記載の再生装置。

10      10. 前記デコードにより得られた非圧縮グラフィクスは、コード値を用いて表現され、

前記再生装置は、

10      グラフィクスを表すコード値を、画素値に変換するルックアップテーブル部を備え、

制御情報は、パレットデータと組みになって、記録媒体から読み出され、

前記グラフィクスデコードによる表示とは、

15      記録媒体から読み出されたパレットデータを、ルックアップテーブル部に設定し、そのパレットデータを用いた画素値変換を、ルックアップテーブル部に命じることである

ことを特徴とする請求項 7 記載の再生装置。

20      11. グラフィクスデコードによる表示は、既に表示されているグラフィクスの更新であり、

20      制御情報は、更新フラグを有し、

更新フラグが所定の値であるなら、ルックアップテーブル部に対するパレットデータの設定のみを行うことで、グラフィクスの更新を行う、ことを特徴とする請求項 10 記載の再生装置。

25      12. 前記グラフィクスストリームはウィンドウ情報を有し、

25      ウィンドウ情報は、各ピクチャを表示するための画面の一部を、グラフィクス表示用のウィンドウとして指定する情報であり、

グラフィクスデコードによるグラフィクス描画は、

画面においてウィンドウ情報に示されるウィンドウをクリアする処理、及び、画面におけるウィンドウに対しグラフィクスを書き込む

30      処理を含むこととなされる、請求項 7 記載の再生装置。

13. 記録媒体の記録方法であって、  
アプリケーションデータを作成するステップと、  
作成したデータを記録媒体に記録するステップとを有し、  
前記アプリケーションデータは、

- 5 動画ストリームとグラフィクスストリームとを多重化することにより得られたデジタルストリームを含み、  
グラフィクスストリームは、パケットの配列であり、  
パケットには、グラフィクスデータを格納したものと、制御情報を格納したものとがあり、
- 10 制御情報は、パケット列において自身より前方に存在するグラフィクスデータを、所定のタイミングで動画ストリームと合成して表示する旨を示す、  
ことを特徴とする記録方法。

14. ビデオストリーム、グラフィクスストリームが多重化された  
15 デジタルストリームの再生をコンピュータに行わせるプログラムであって、

ビデオストリームをデコードして動画像を得るステップと、  
グラフィクスストリームをデコードしてグラフィクスを得て、動画像に合成するステップとを備え、

- 20 グラフィクスを得るステップは、  
記録媒体から新たな制御情報が記録媒体から読み込まれれば、その制御情報に基づき、前もって送り込まれたグラフィクスデータを表示する

処理をコンピュータに行わせることを特徴とするプログラム。

- 25 15. ビデオストリーム、グラフィクスストリームが多重化されたデジタルストリームの再生をコンピュータに行わせる再生方法であって、

ビデオストリームをデコードして動画像を得るステップと、  
グラフィクスストリームをデコードしてグラフィクスを得て、動画

- 30 像に合成するステップとを備え、

グラフィクスを得るステップは、

記録媒体から新たな制御情報が記録媒体から読み込まれれば、その制御情報に基づき、前もって送り込まれたグラフィクスデータを表示する

5      ことを特徴とする再生方法。

図 1

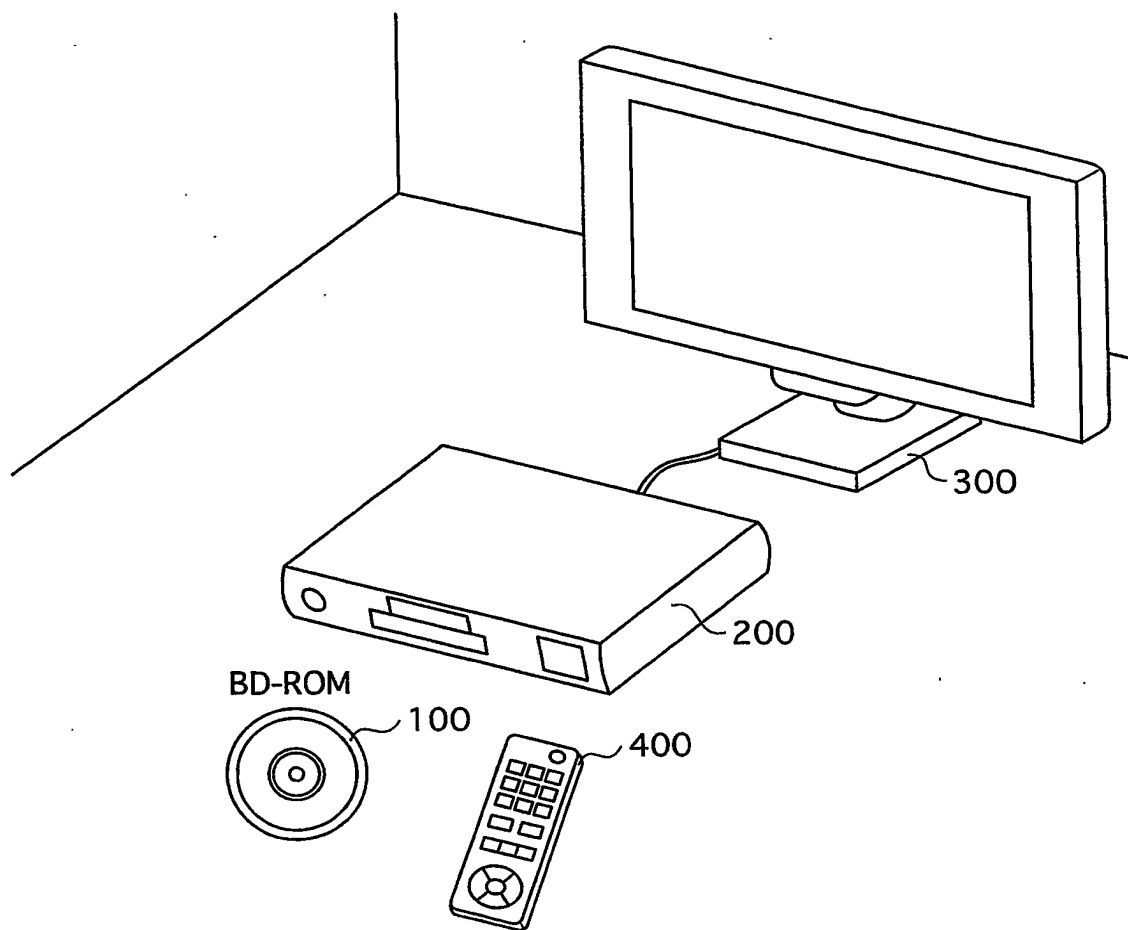


図2

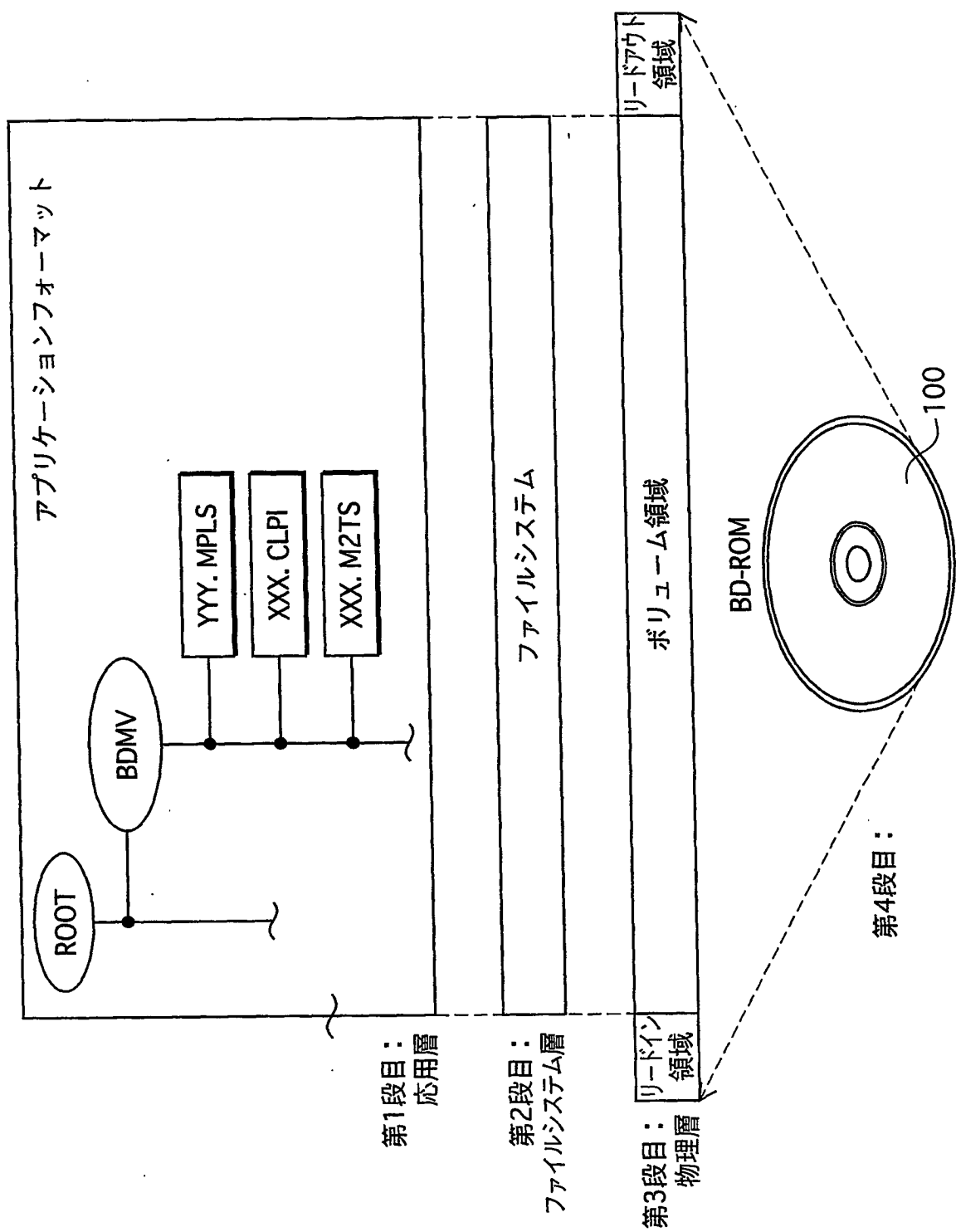




図3

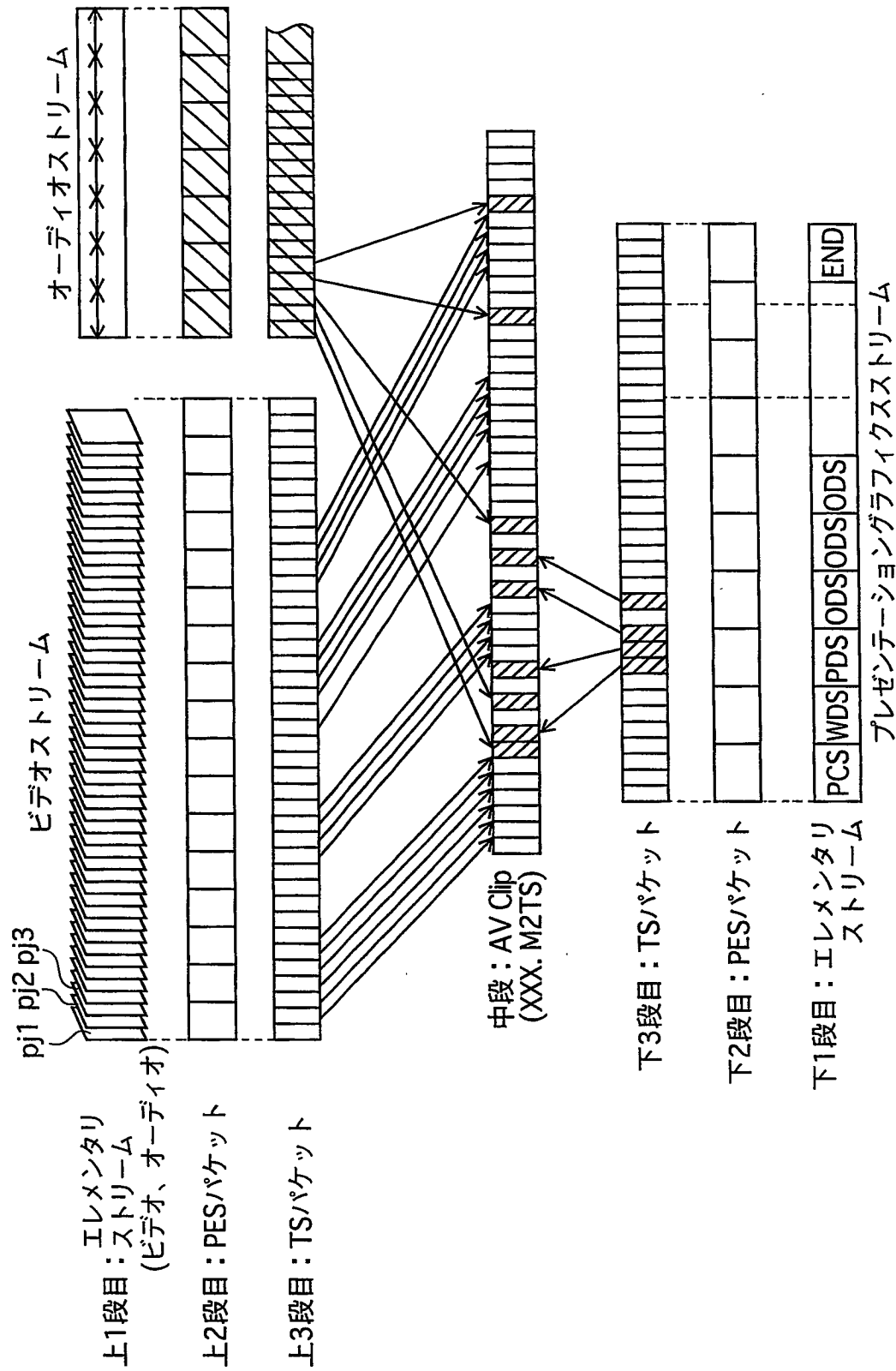


図4

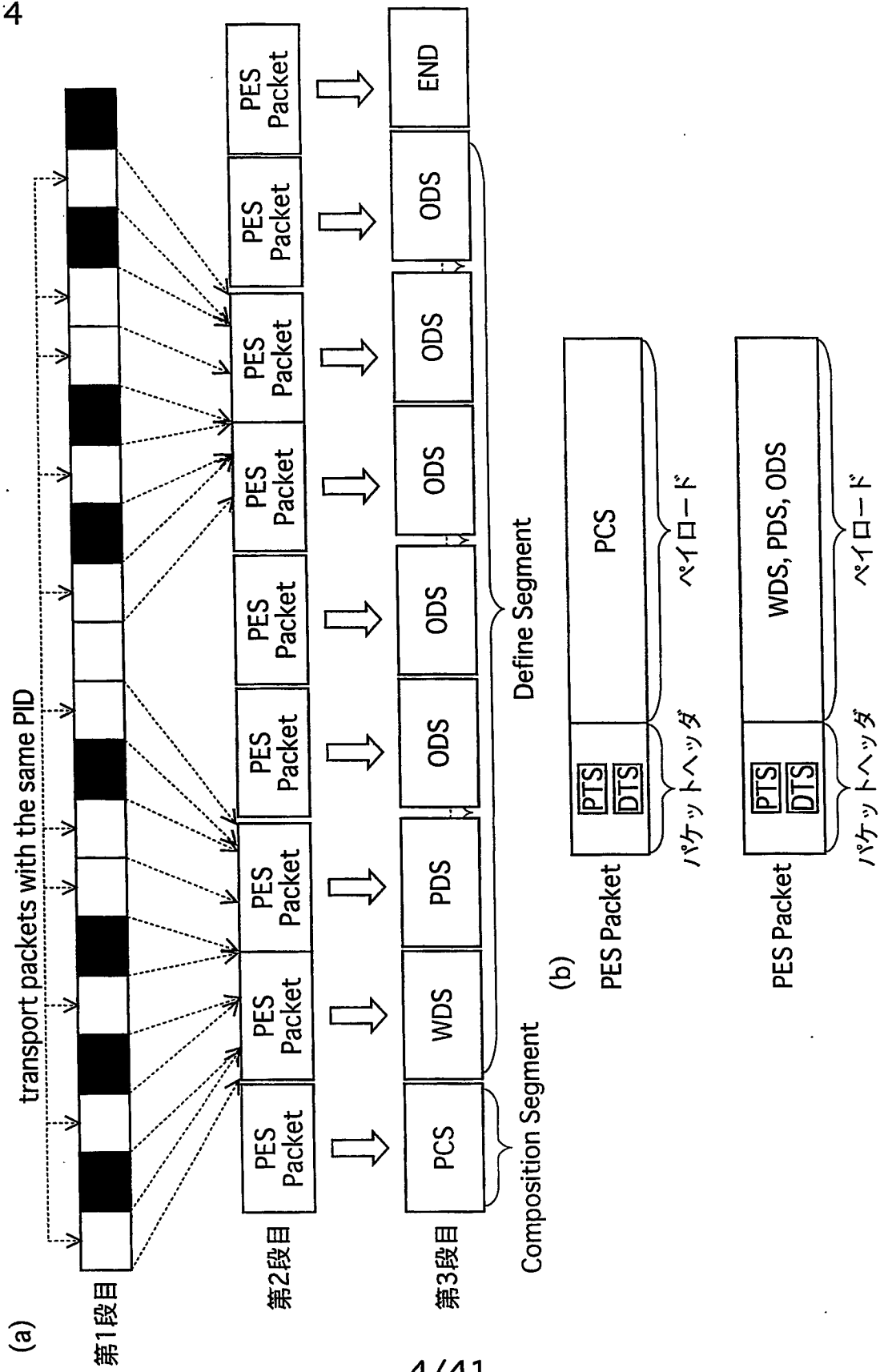


図5

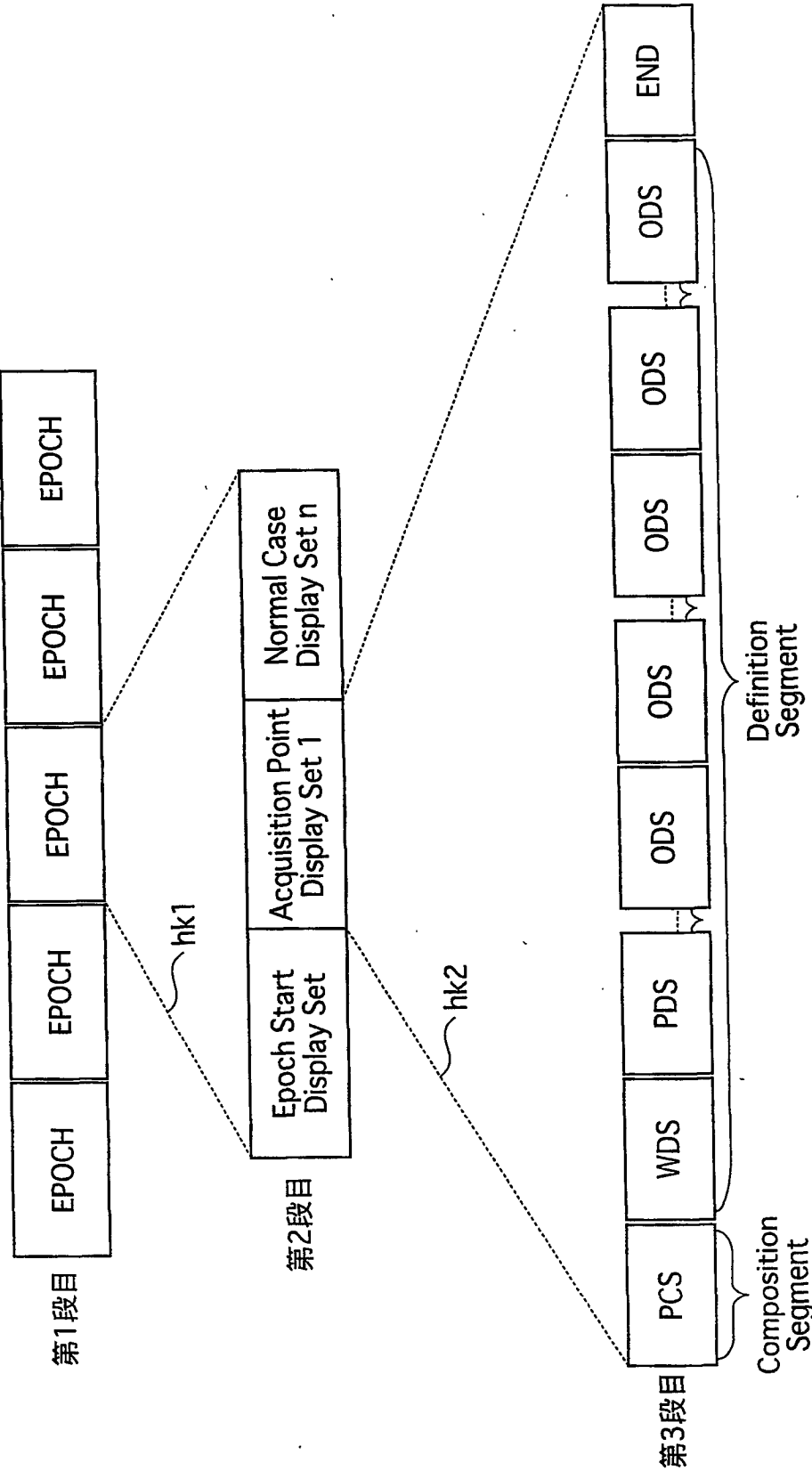


図6

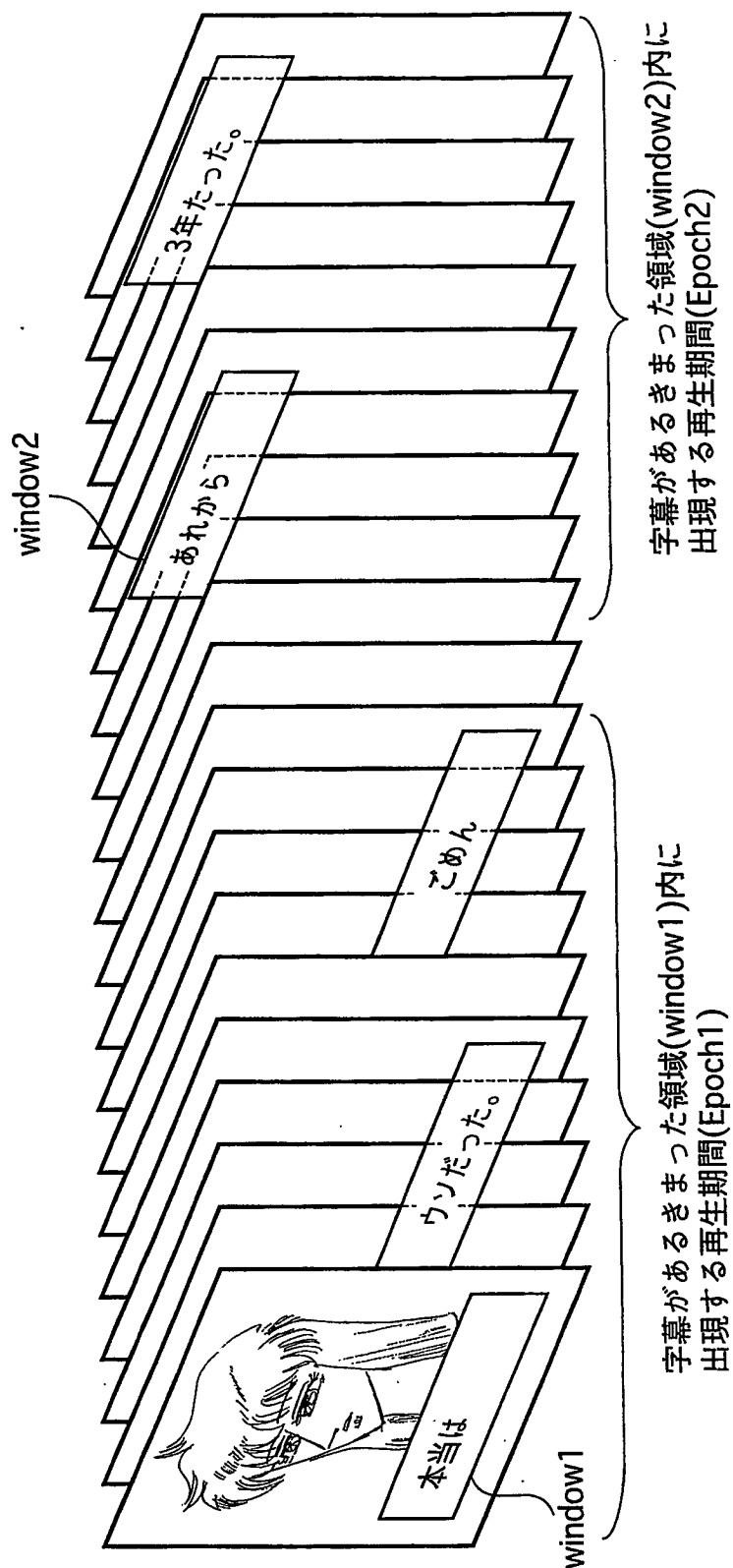


図7

(a)

object\_definition\_segment

segment_type
segment_length
object_id
object_version_number
last in sequence flag
object_data_fragment

圧縮された  
グラフィクスオブジェクト

(b)

palette\_definition\_segment

segment_type
segment_length
palette id
palette version_number
palette_entry

Y_value
Cr_value
Cb_value
T_value

図8

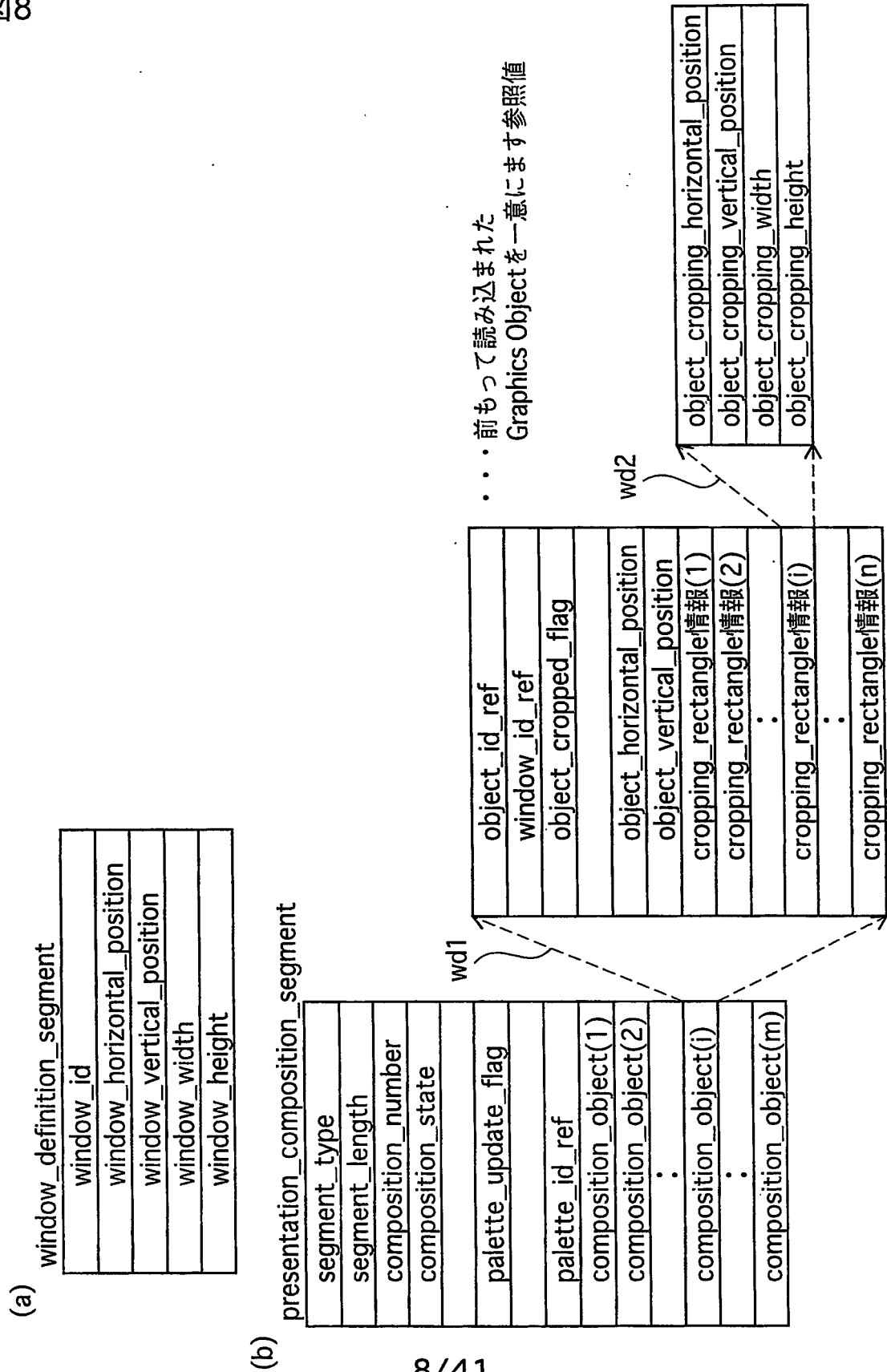


図9

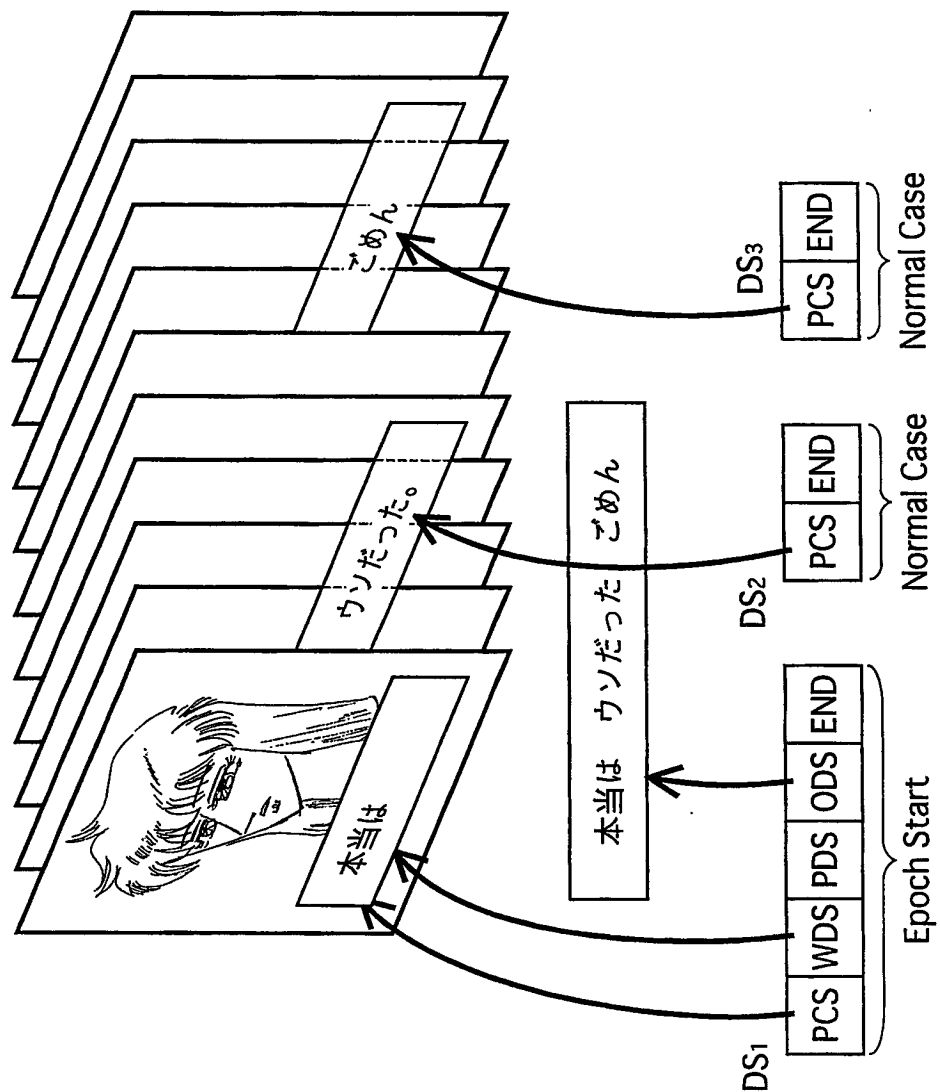


図10

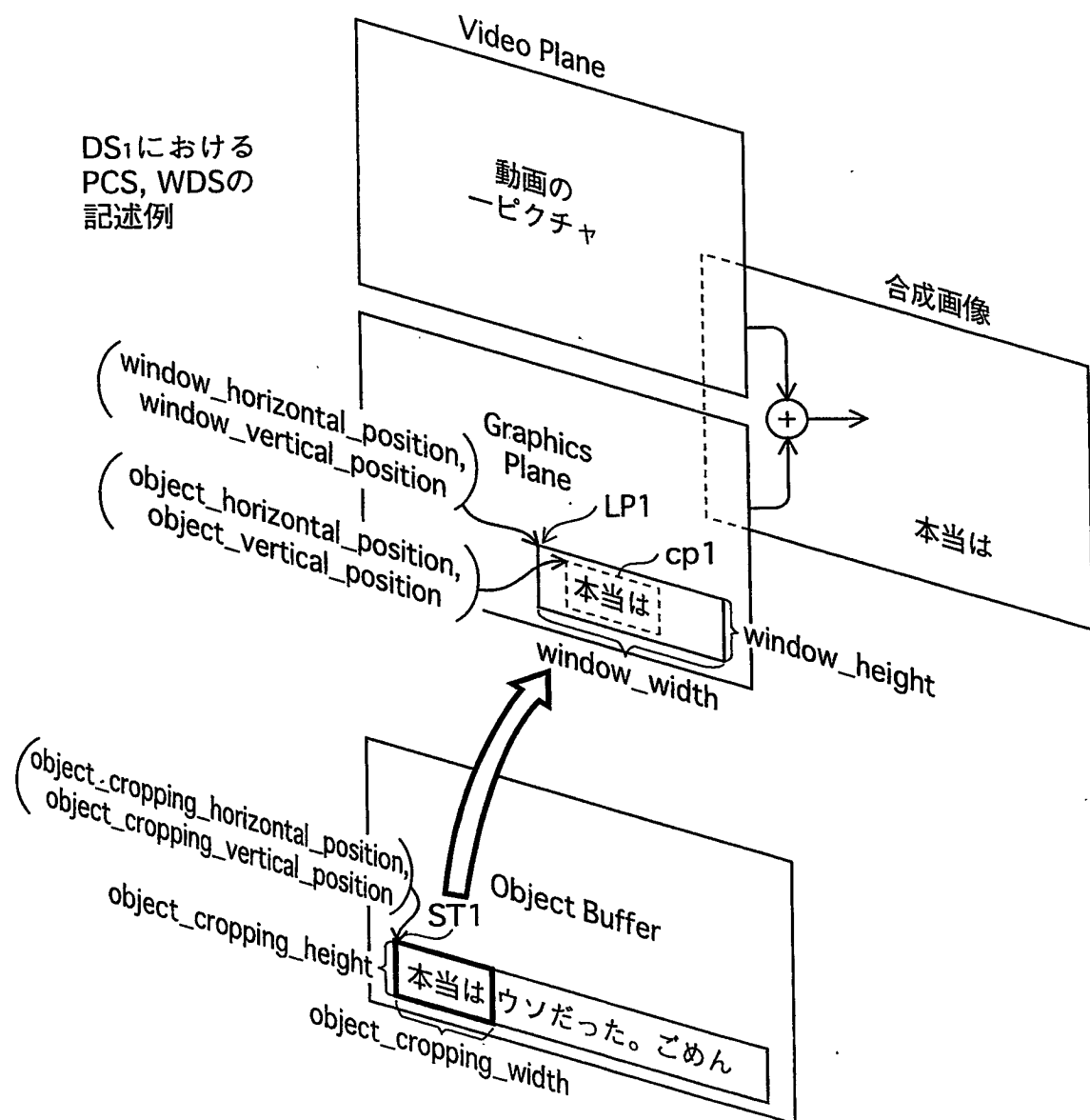




図11

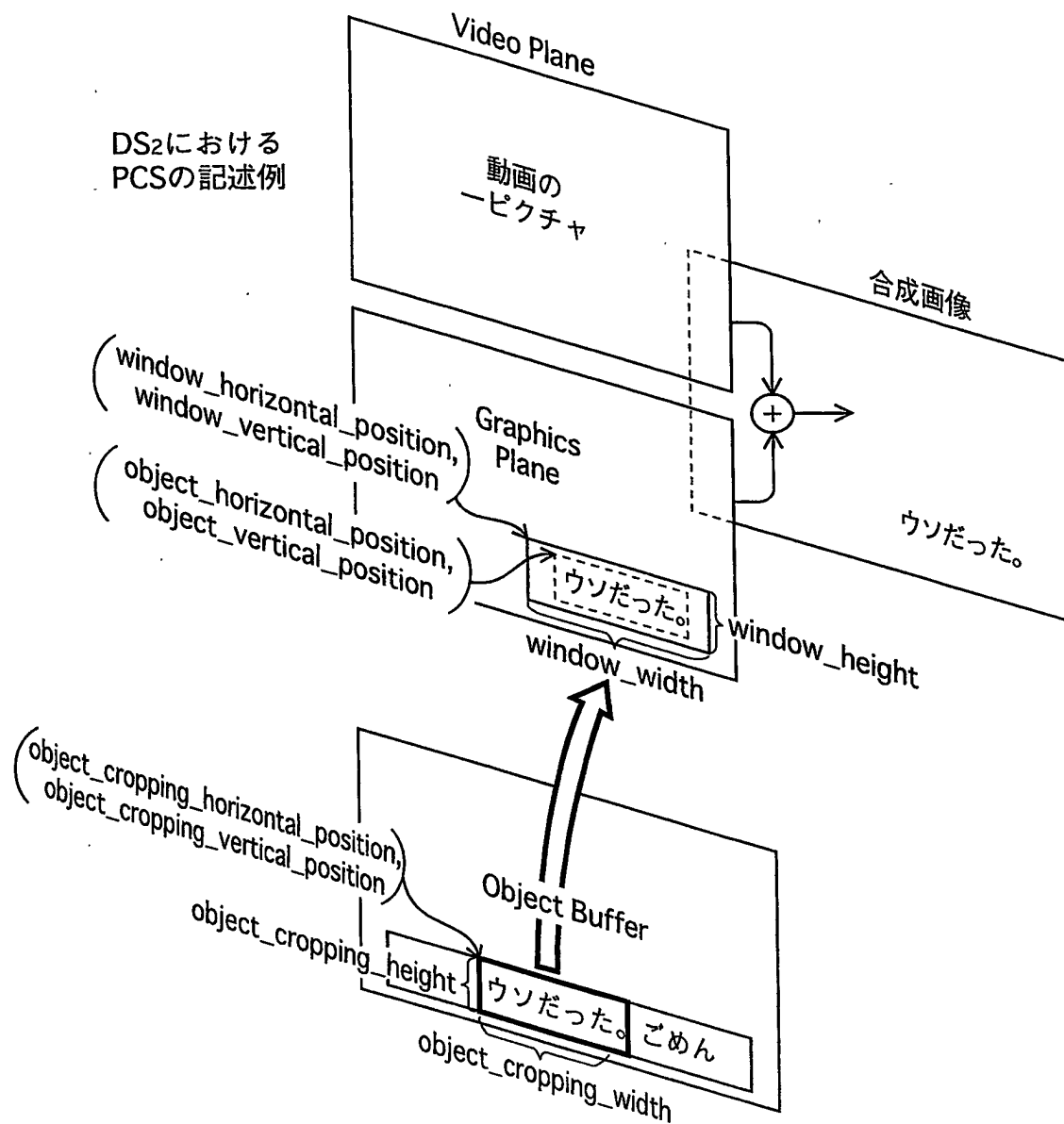


図12

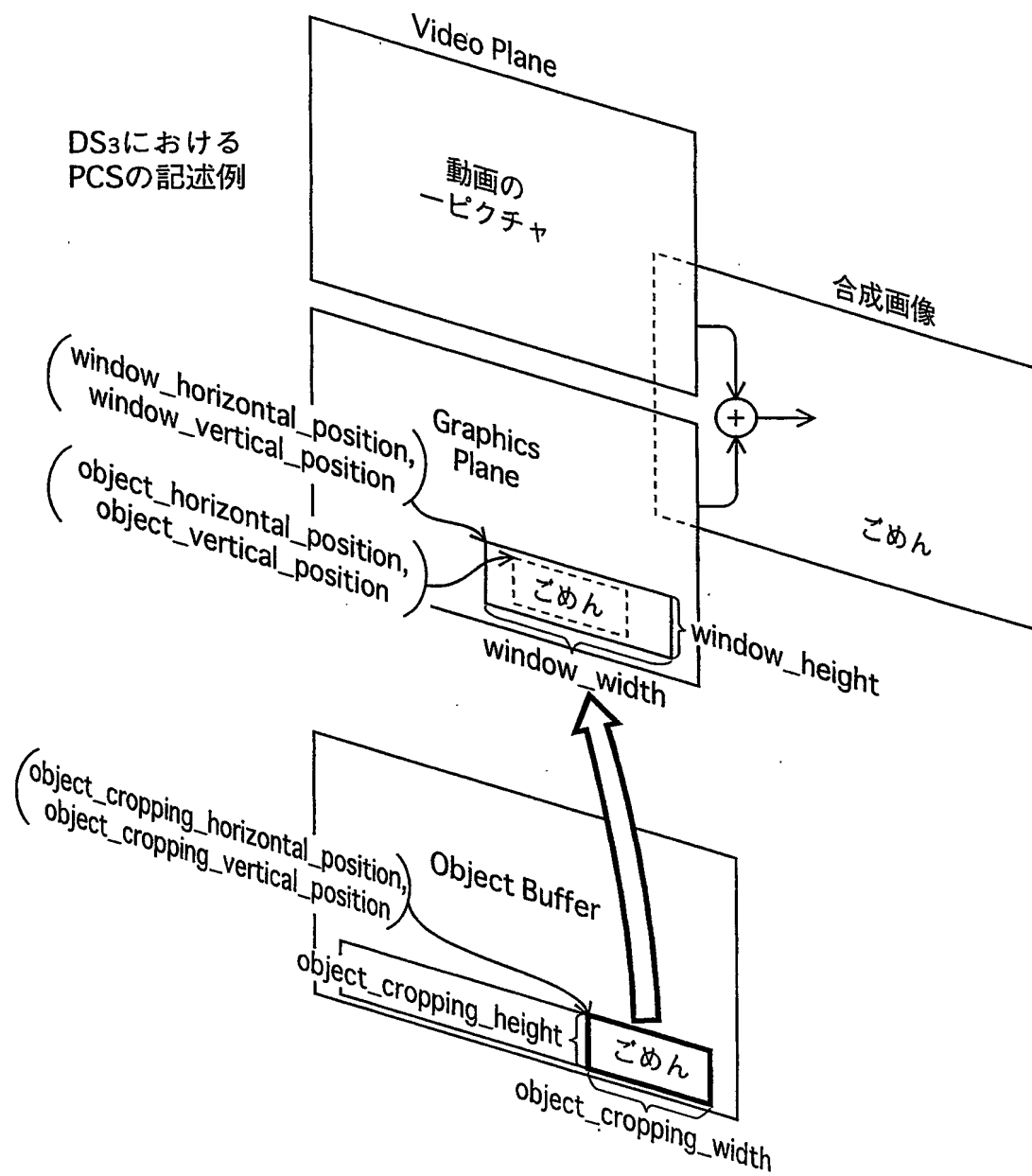
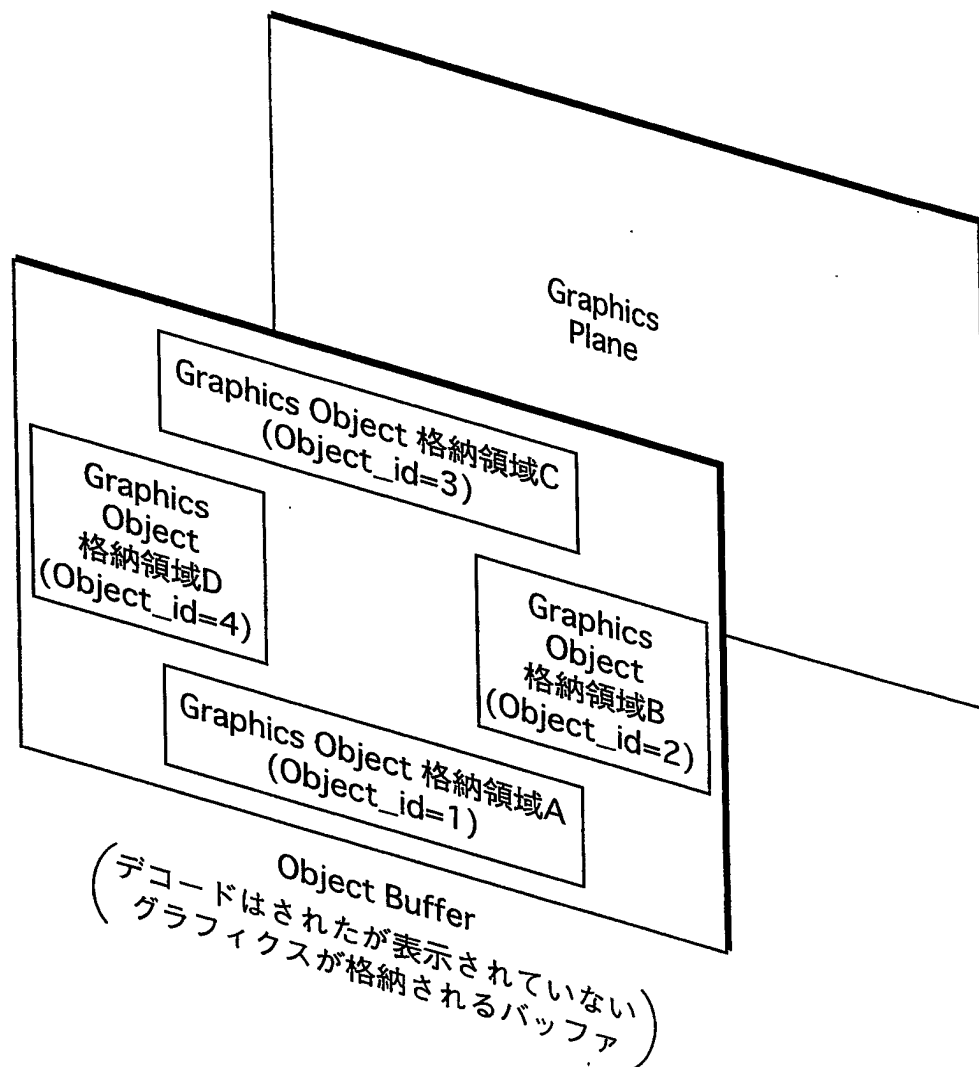


図13



14  $\text{PTS}(\text{DSn}[\text{PCS}]) \geq \text{DTS}(\text{DSn}[\text{PCS}]) + \text{DECODEDURATION}(\text{DSn})$

Where:

- $\text{DECODEDURATION}(\text{DSn})$  is calculated as follows:

```

decode_duration = 0 ;
decode_duration += PLANEINITIALIZATIONTIME( DSn ) ;
if( DSn. PCS. num_of_objects == 2 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    if( DSn. PCS. OBJ[0]. window_id == DSn. PCS. OBJ[1]. window_id )
    {
        decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
        decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*106 ) ;
    }
    else
    {
        decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*106 ) ;
        decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
        decode_duration += 90000*( SIZE( DSn. PCS. OBJ[1]. window_id )//256*106 ) ;
    }
}
else if( DSn. PCS. num_of_objects ==1 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*106 ) ;
}
return decode_duration ;

```

- $\text{PLANEINITIALIZATIONTIME}(\text{DSn})$  is calculated as follows:

```

initialize_duration=0 ;
if( DSn. PCS. composition_state== EPOCH_START )
{
    initialize_duration = 90000*( 8*video_width*video_height//256*106 ) ;
}
else
{
    for( i=0 ; i < WDS. num_windows ; i++ )
    {
        if( EMPTY(DSn.WDS.WIN[i],DSn) )
            initialize_duration += 90000*( SIZE( DSn. WDS. WIN[i] )//256*106 ) ;
    }
}
return initialize_duration ;

```

- $\text{WAIT}(\text{DSn}, \text{OBJ}, \text{current\_duration})$  is calculated as follows:

```

wait_duration = 0 ;
if( EXISTS( OBJ. object_id, DSn ) )
{
    object_definition_ready_time = PTS( GET( OBJ. object_id. DSn ) ) ;
    current_time = DTS( DSn. PCS )+current_duration ;
    if( current_time < object_definition_ready_time )
        wait_duration += object_definition_ready_time - current_time ;
}
return wait_duration ;

```

図15

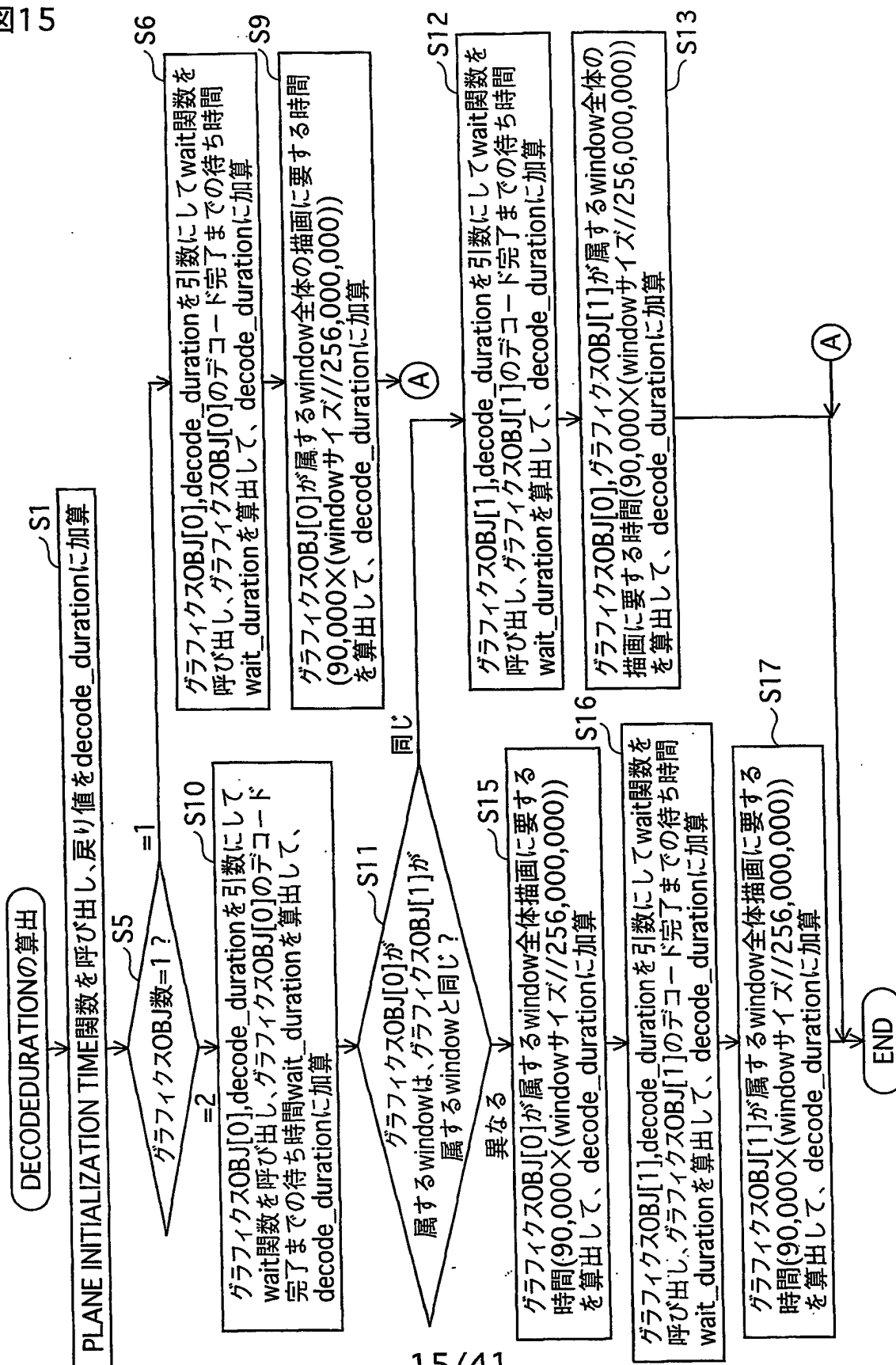
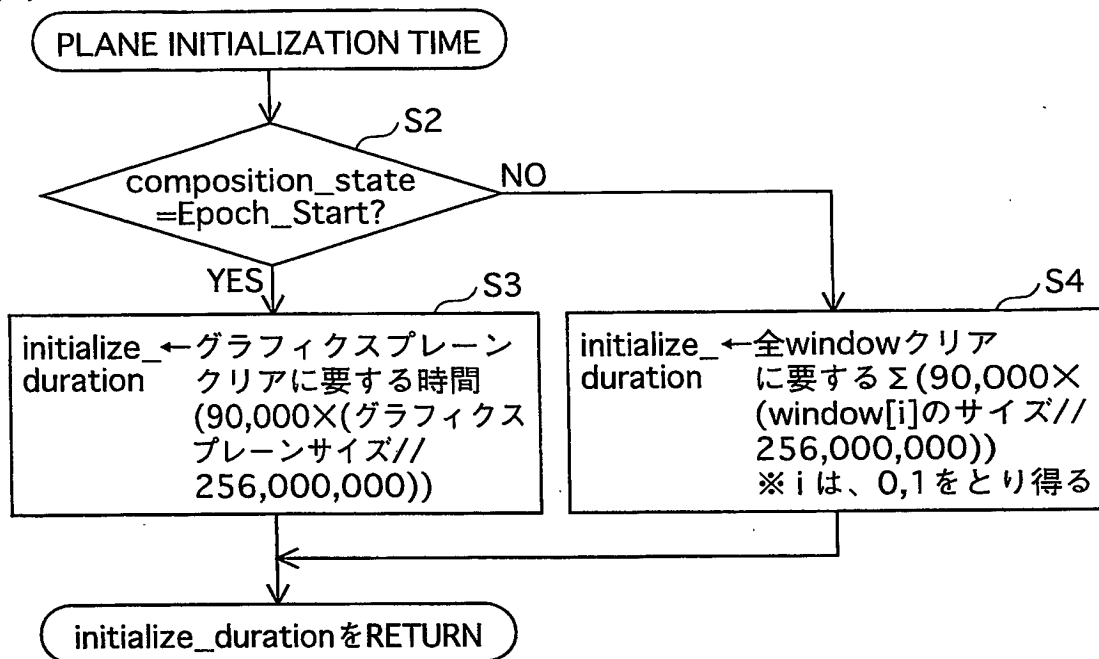


図16

(a)



(b)

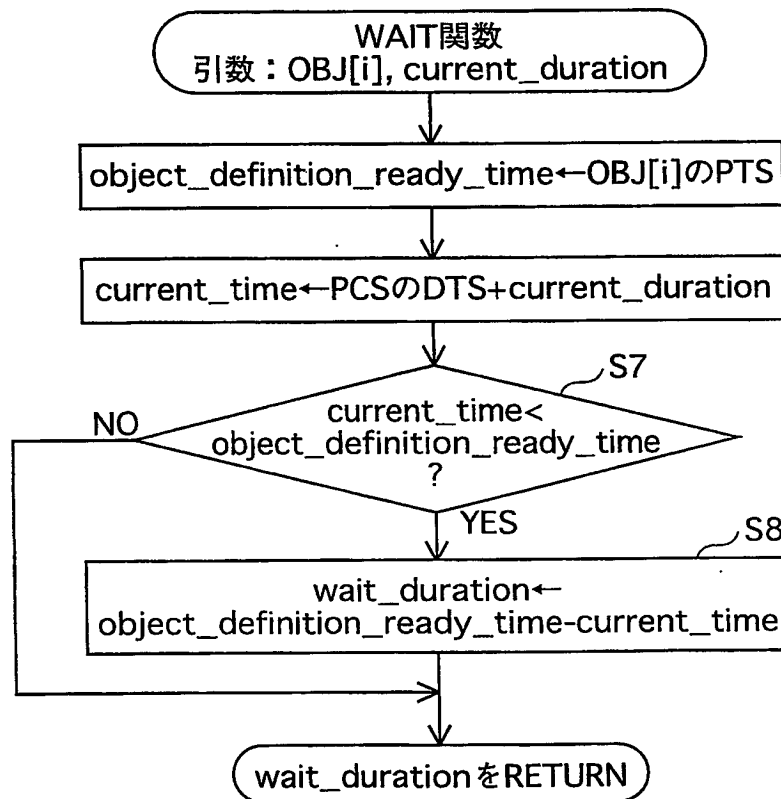


図17

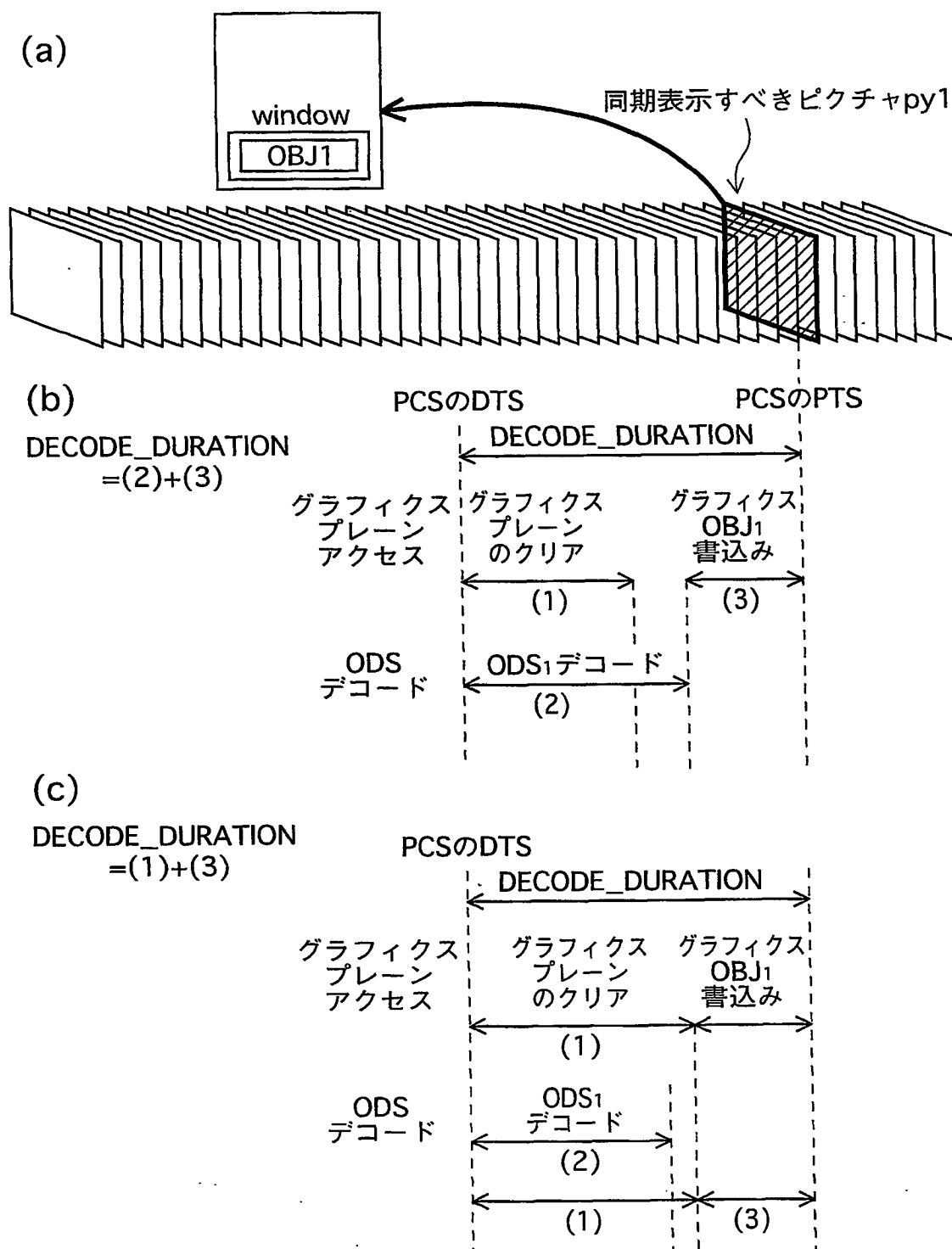
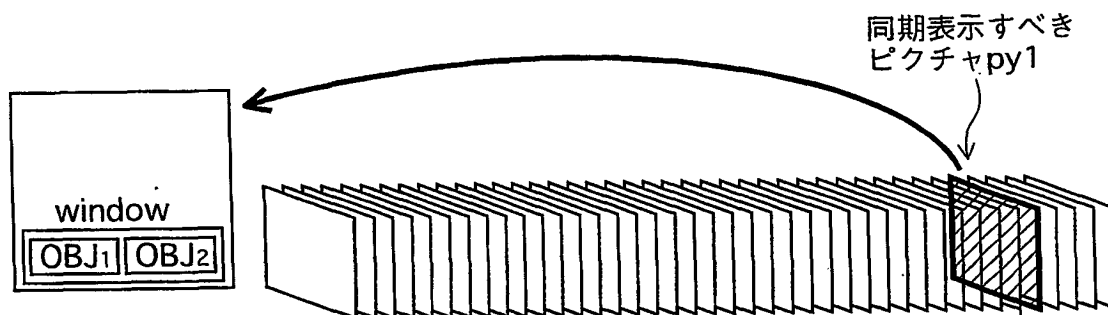
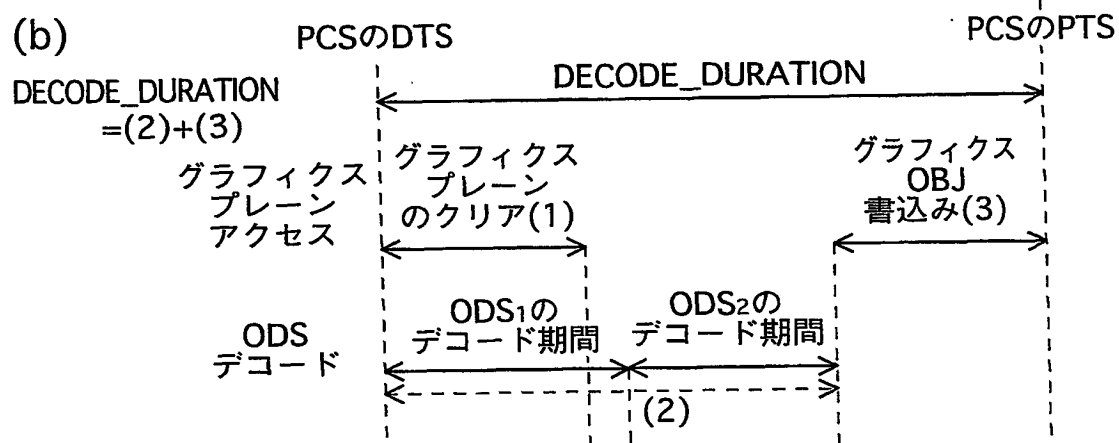


図18

(a)



(b)



(c)

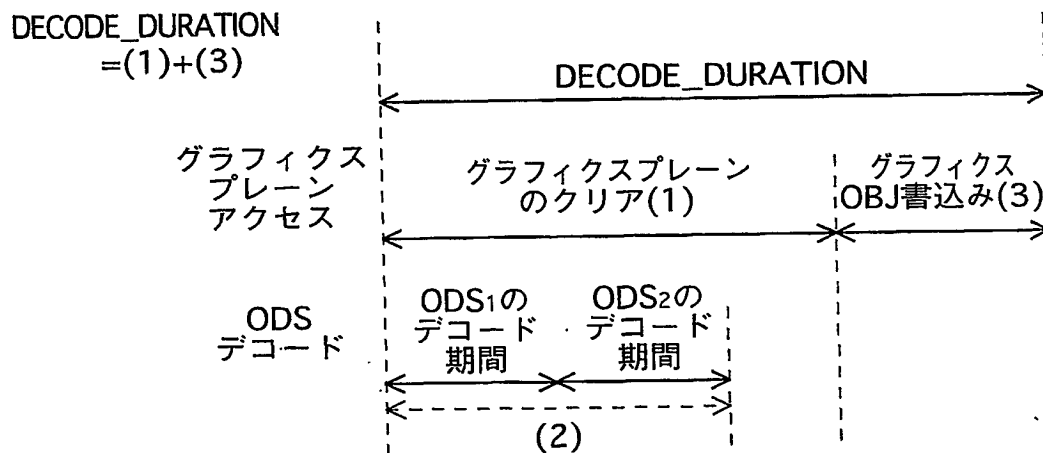




図19

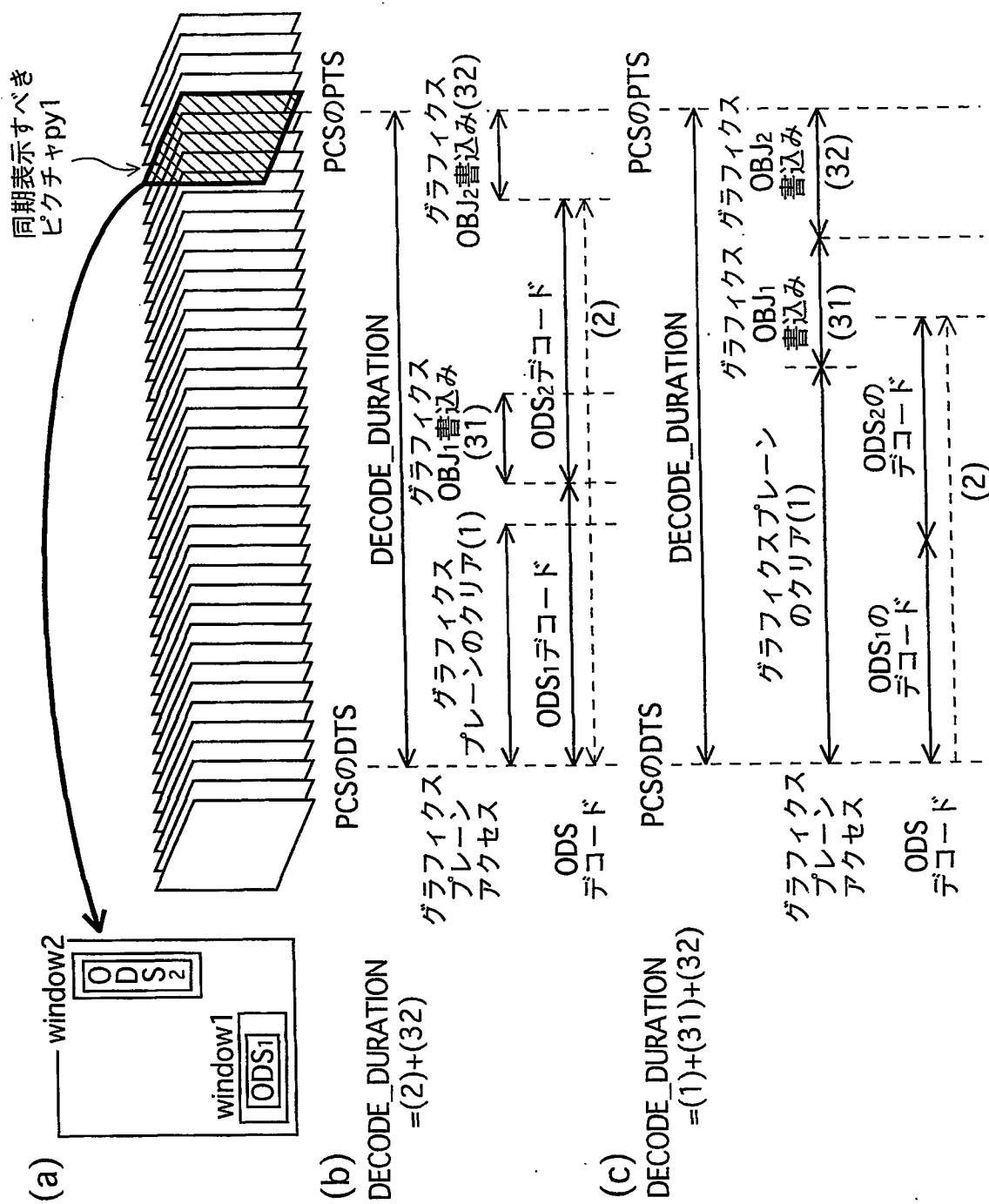


図20

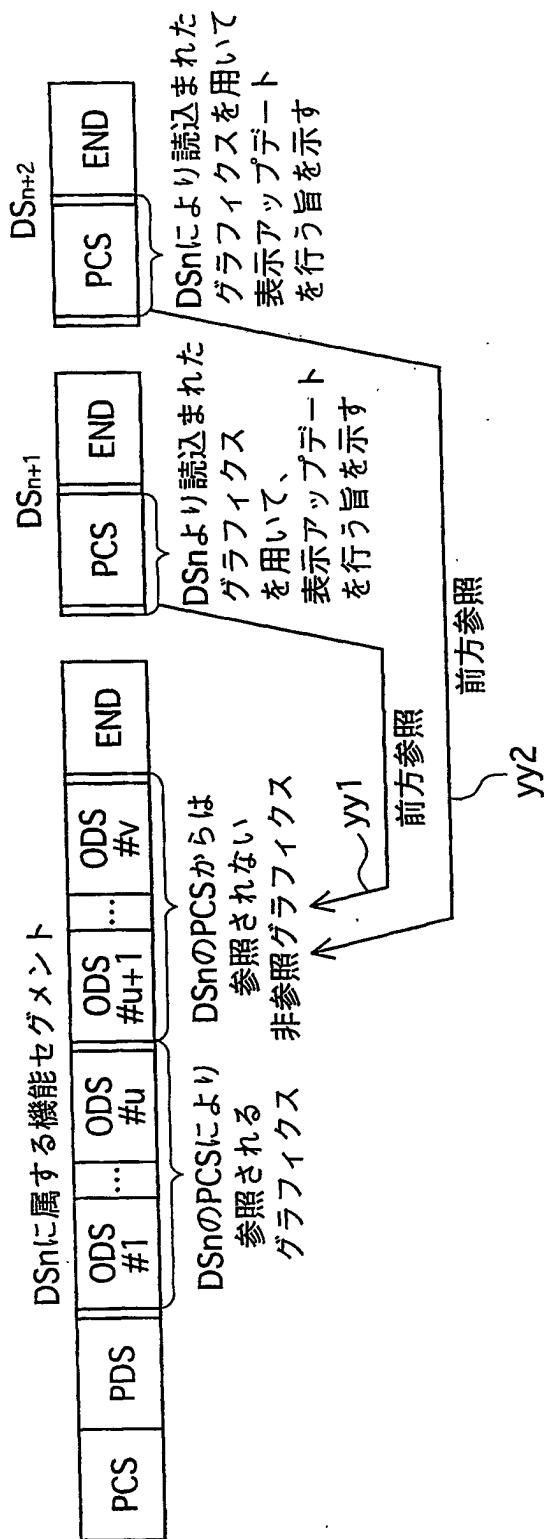


図21

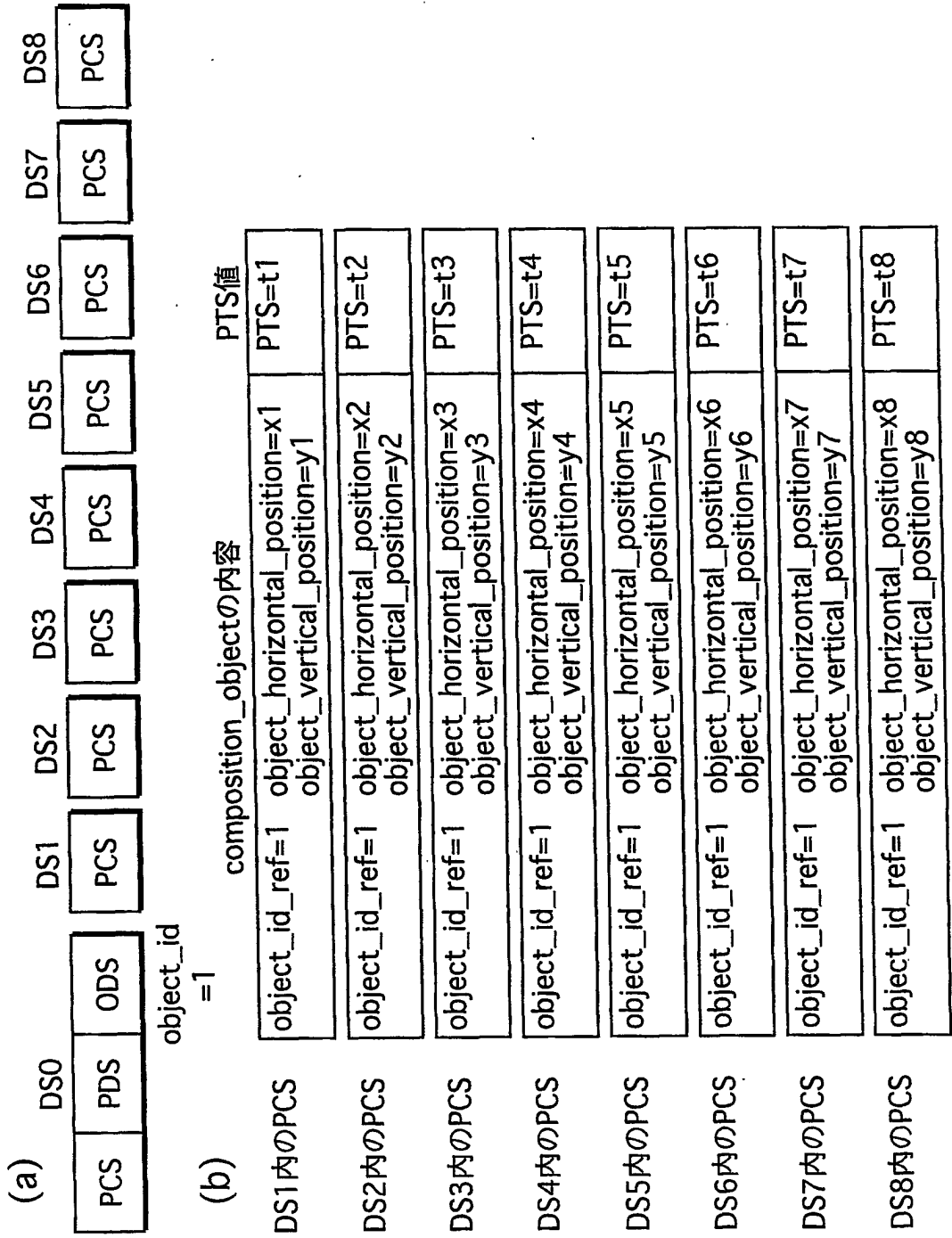


図22

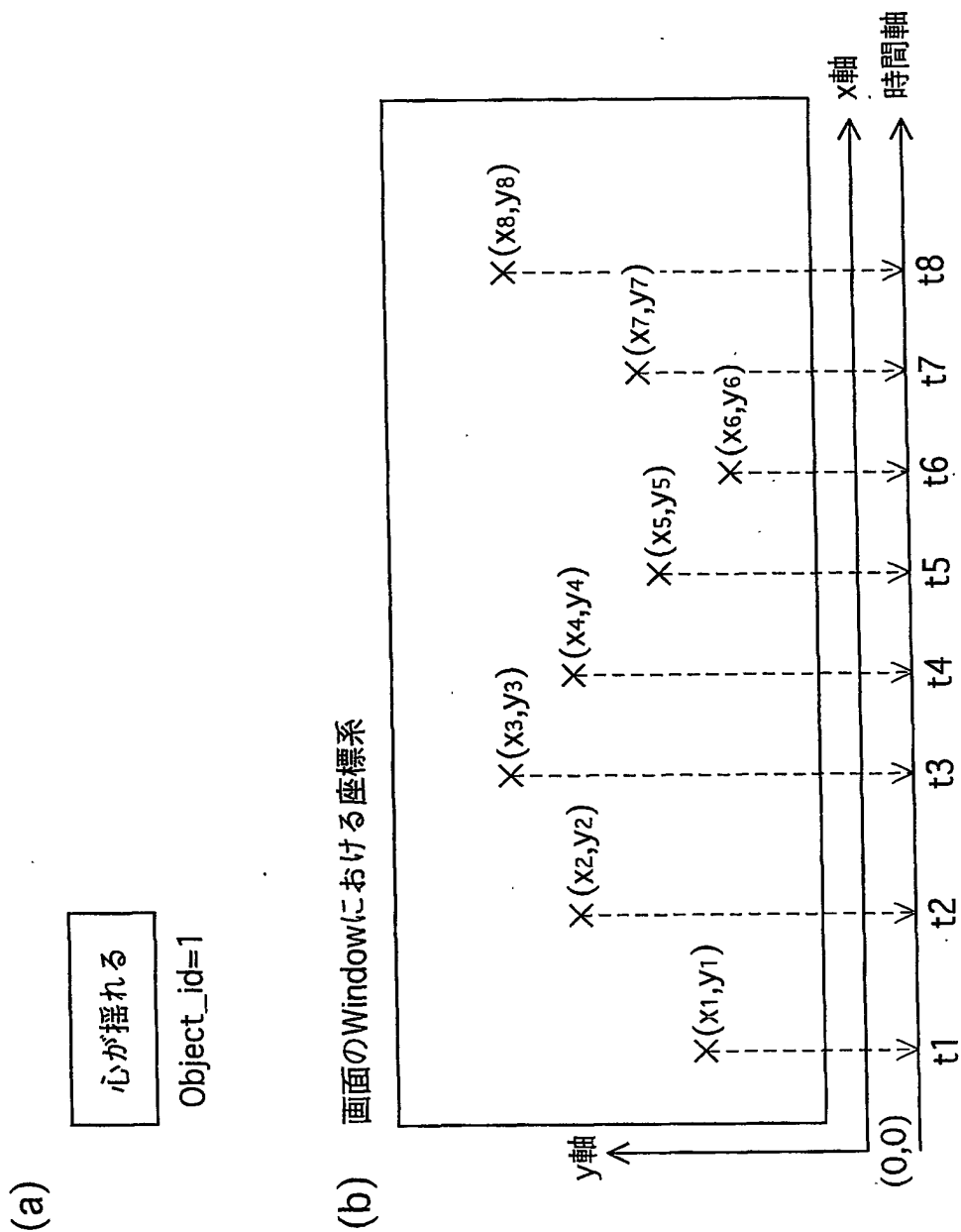


図23

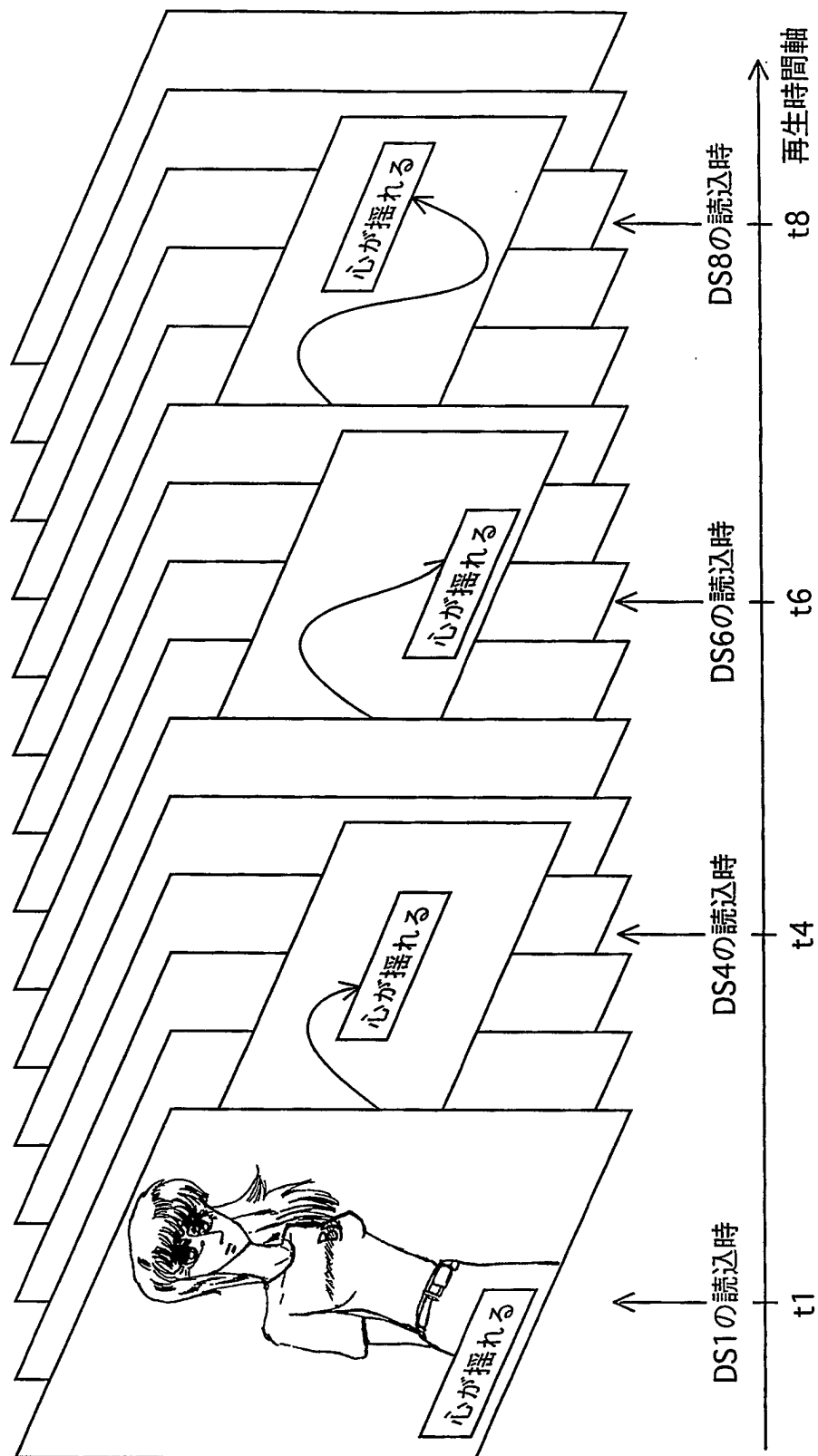


図24

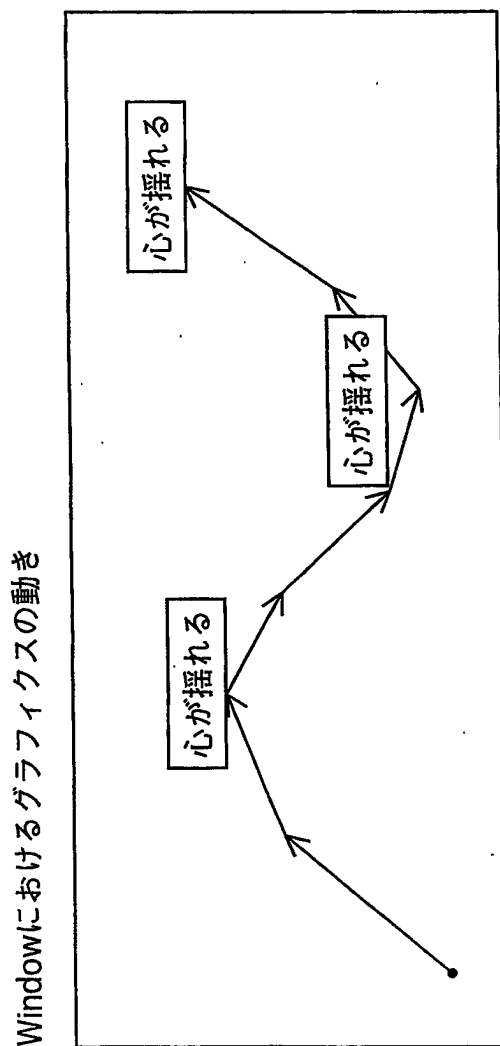


図25

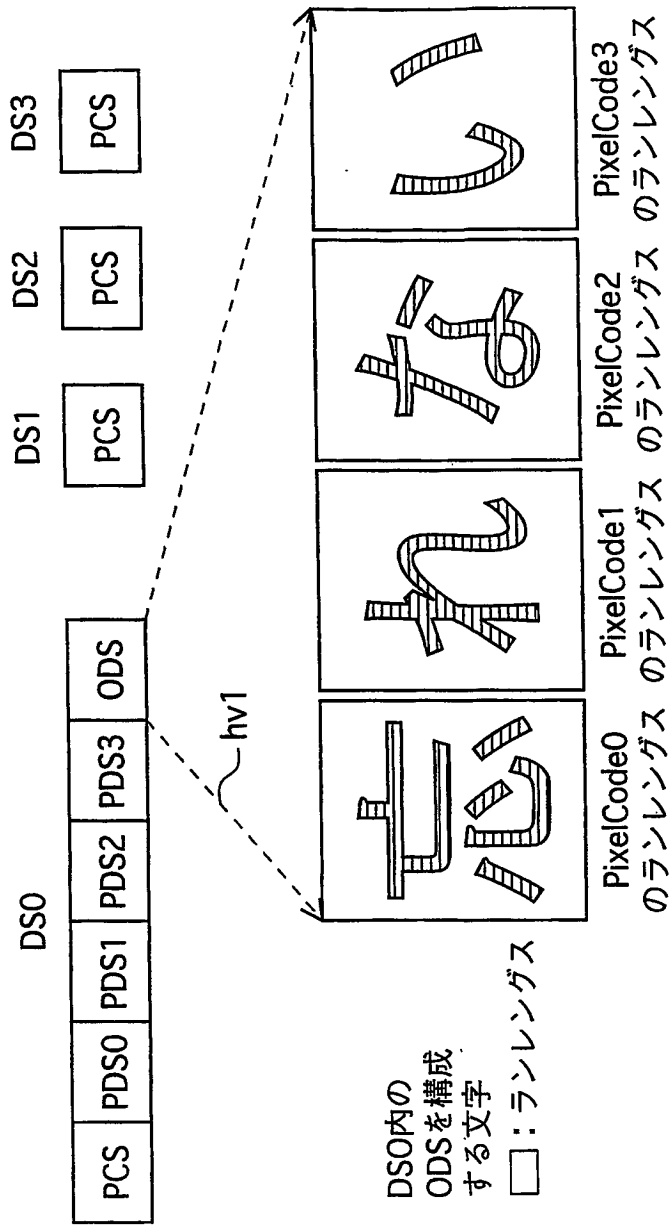


図26

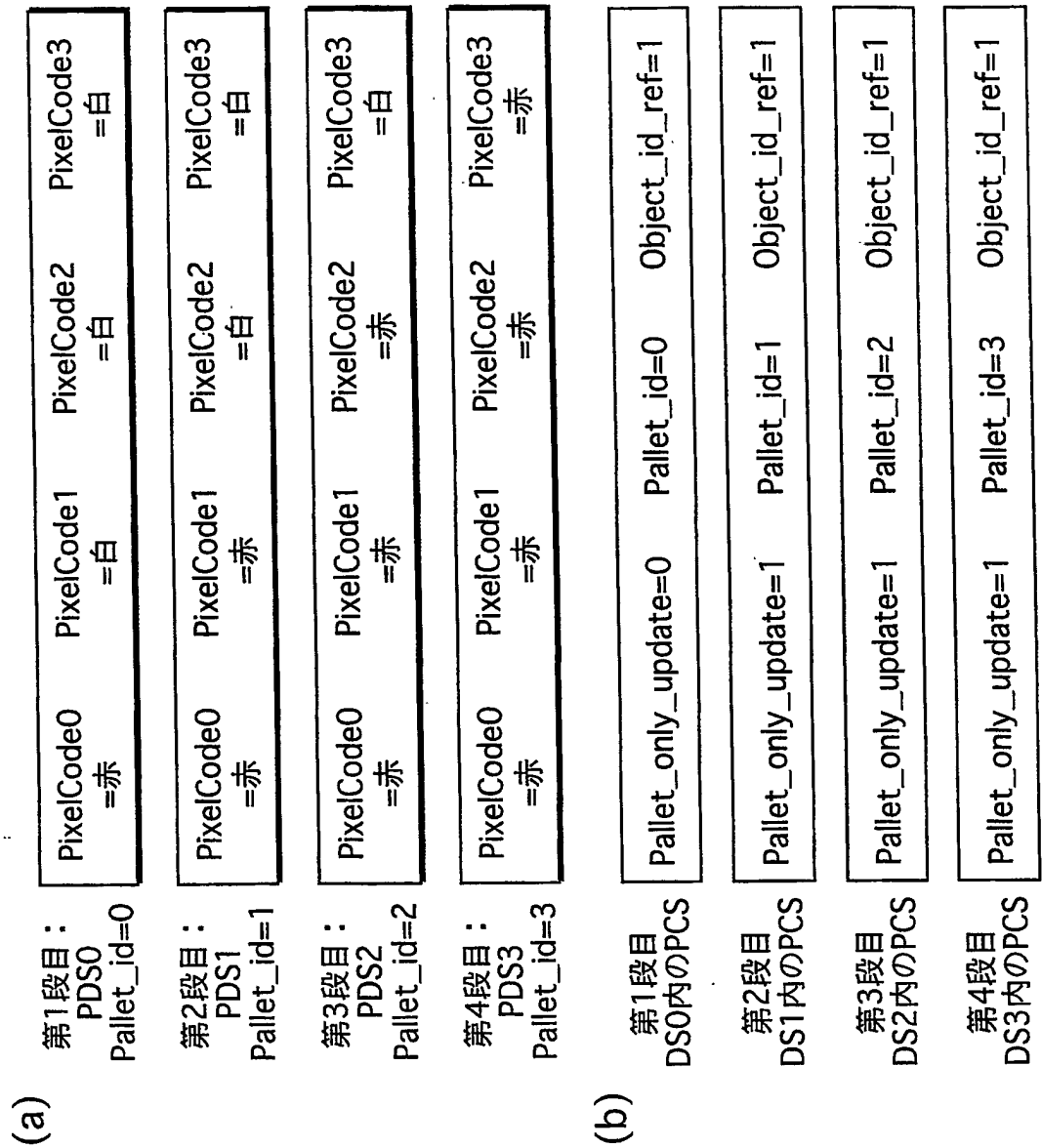




図27

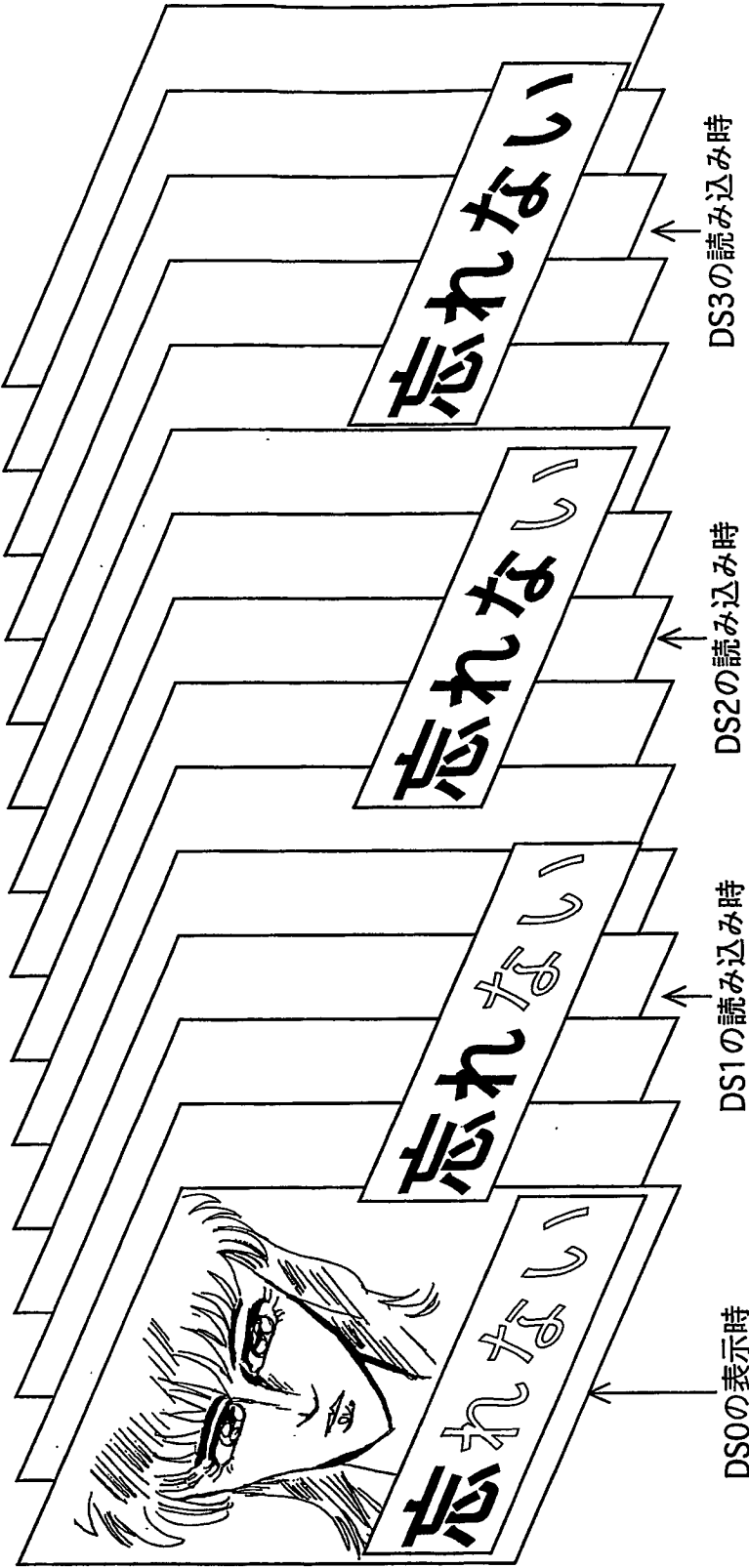


図28

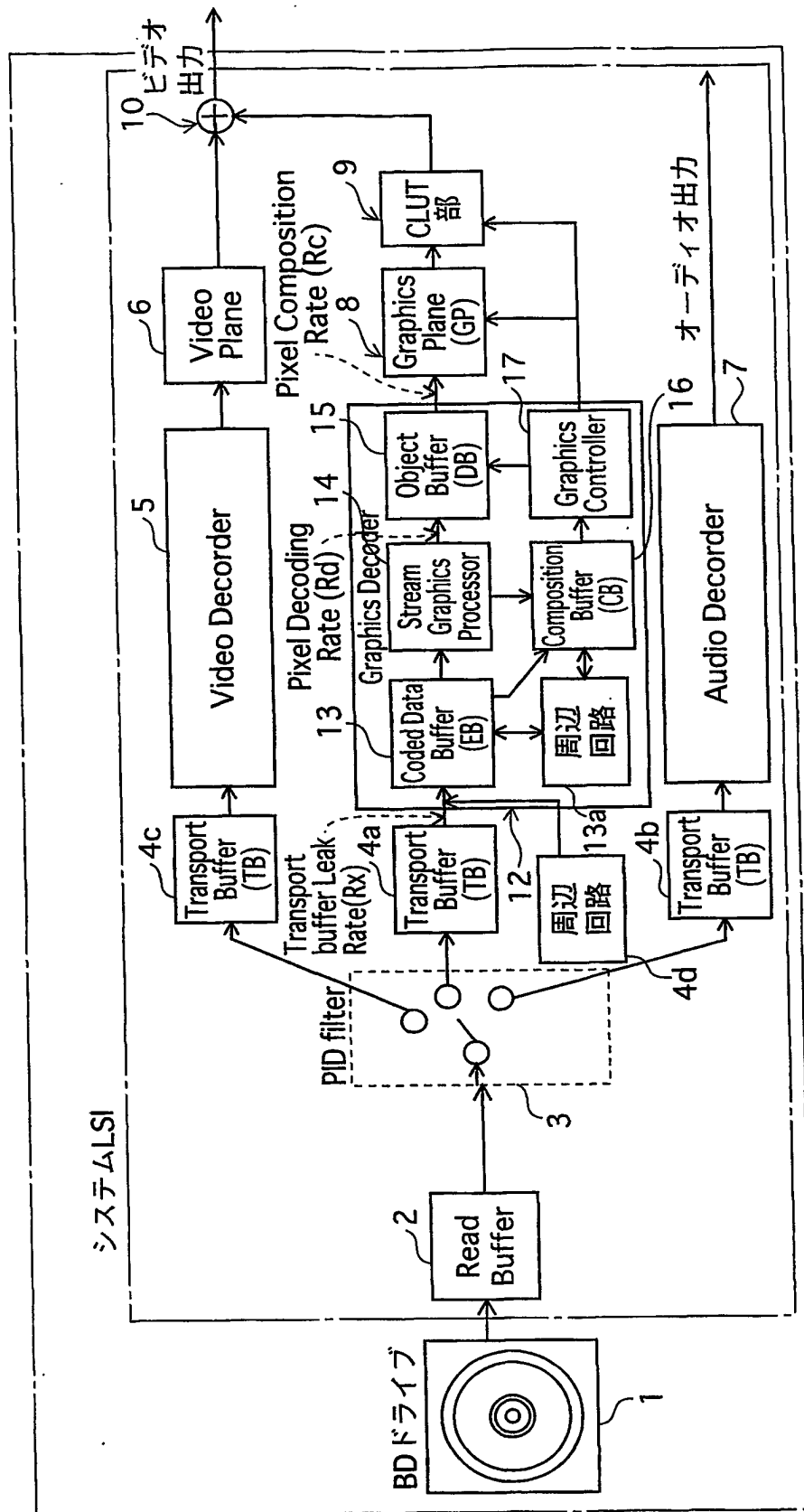


図29

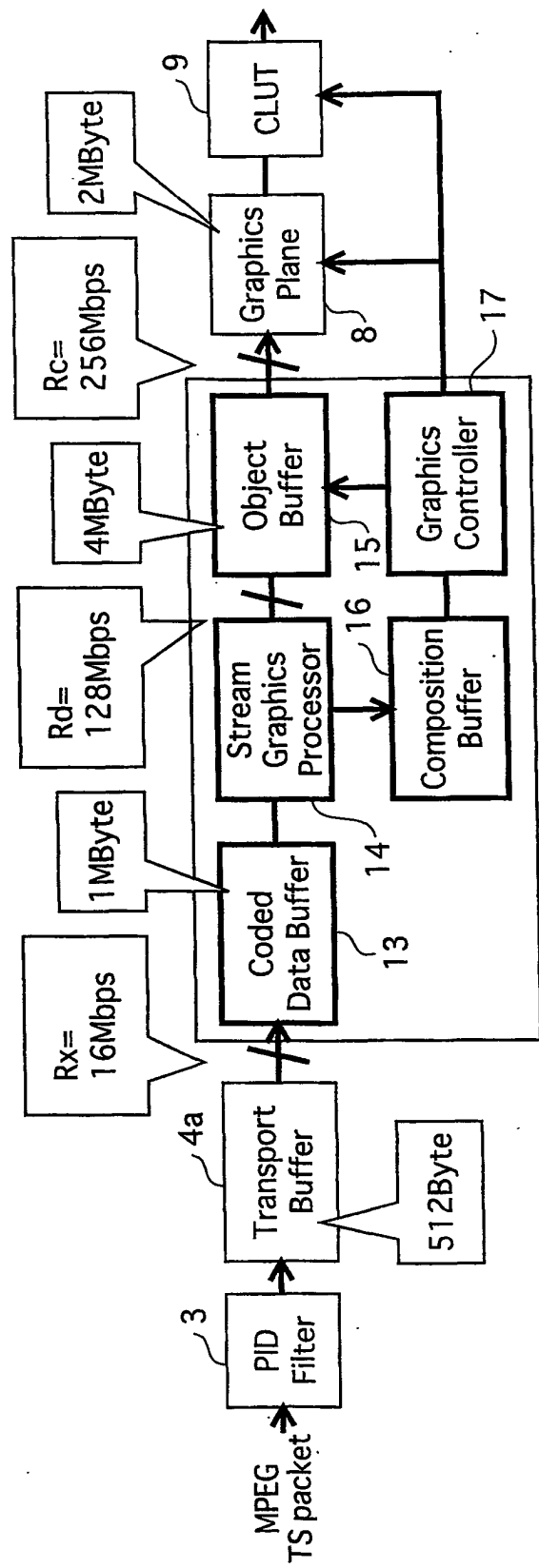


図30

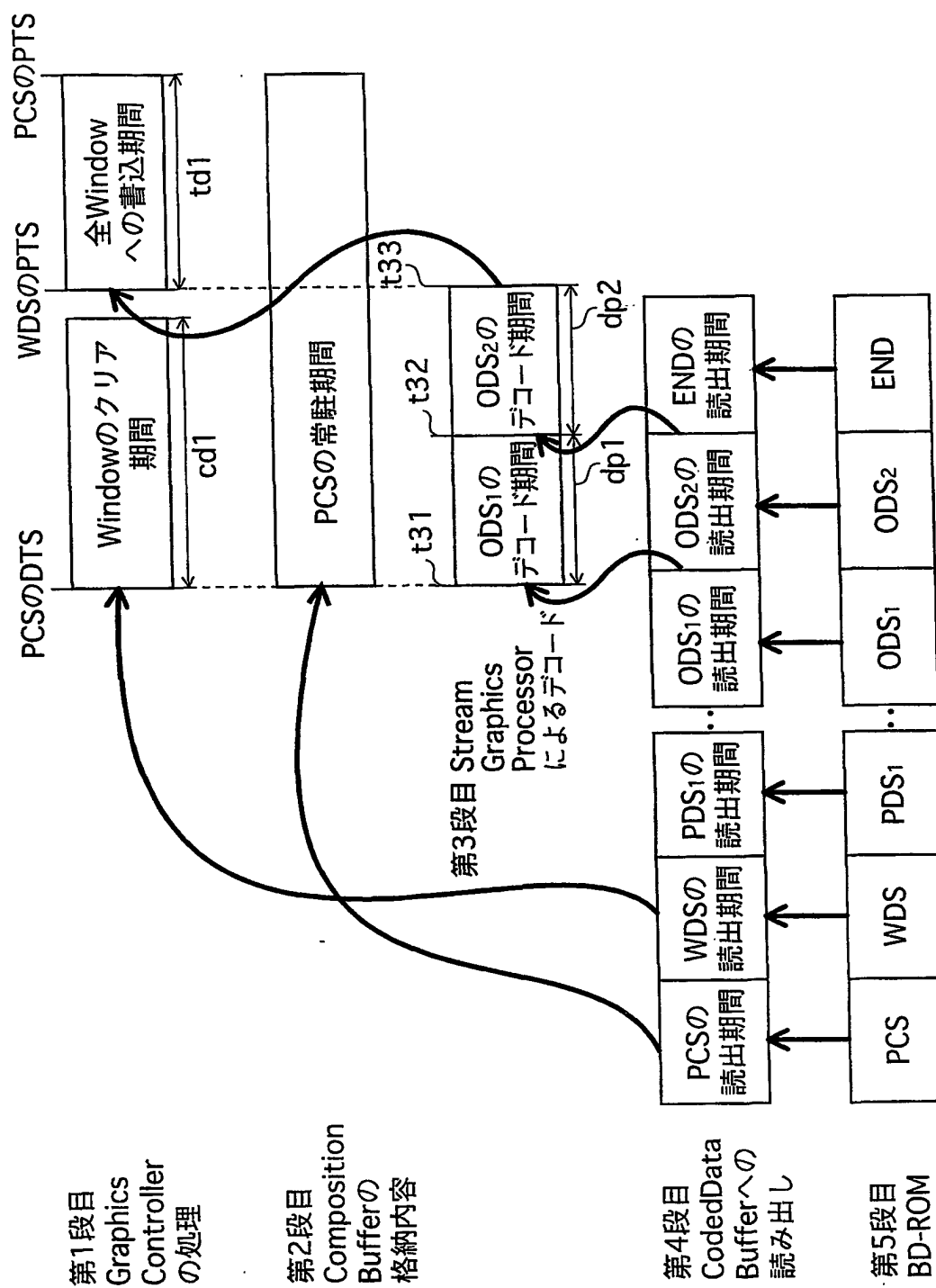


図31

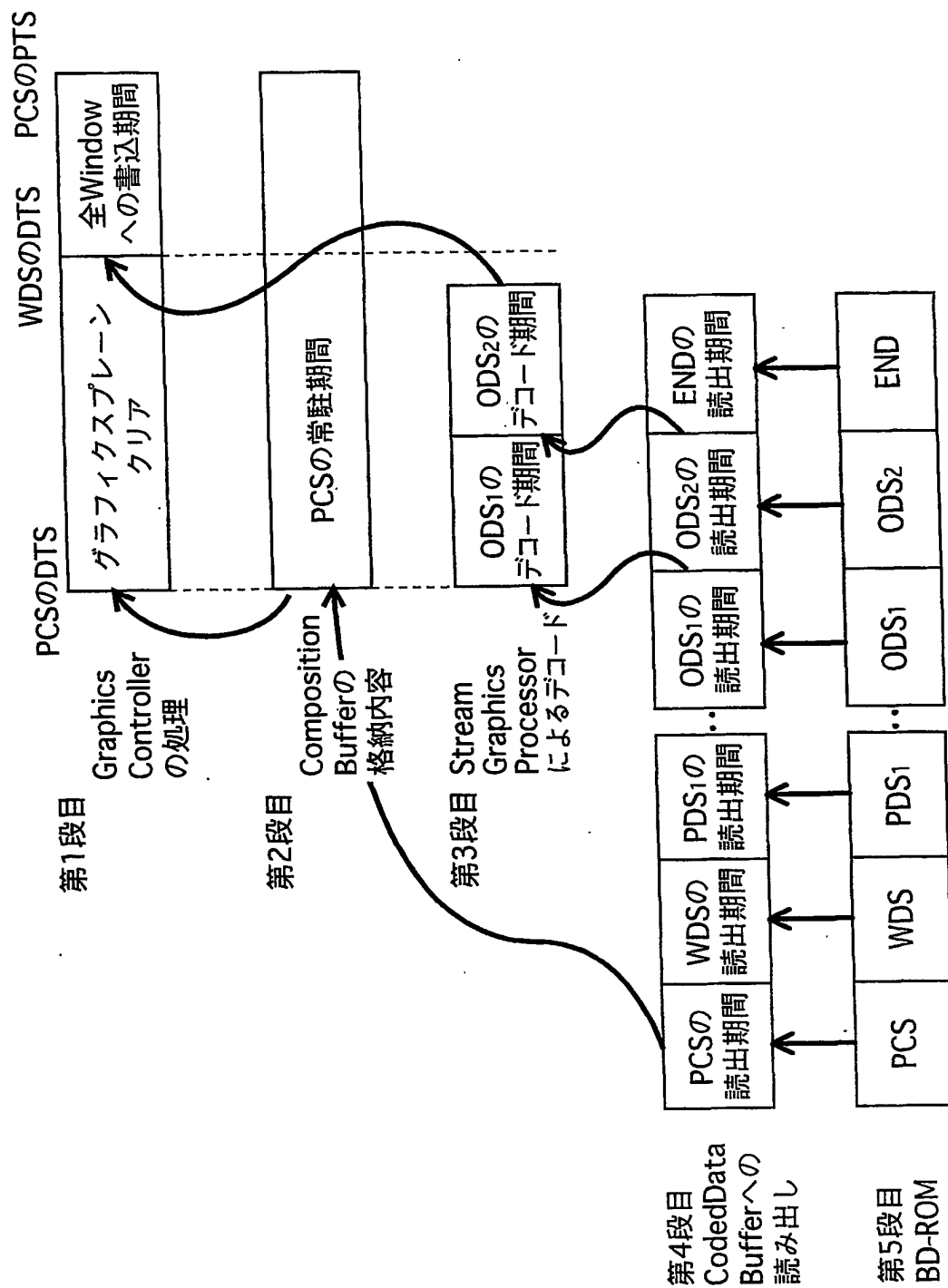


図32

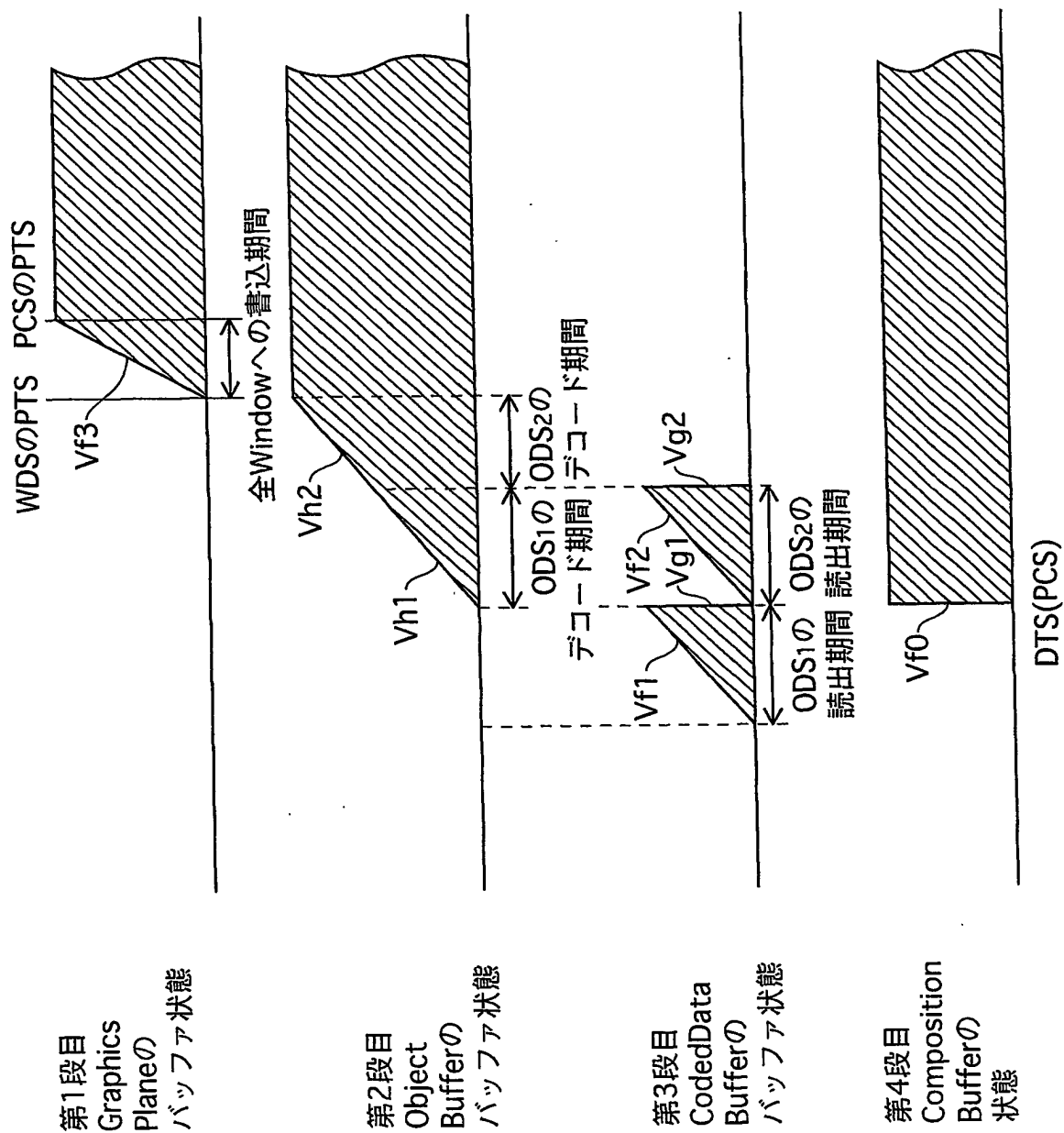


図33

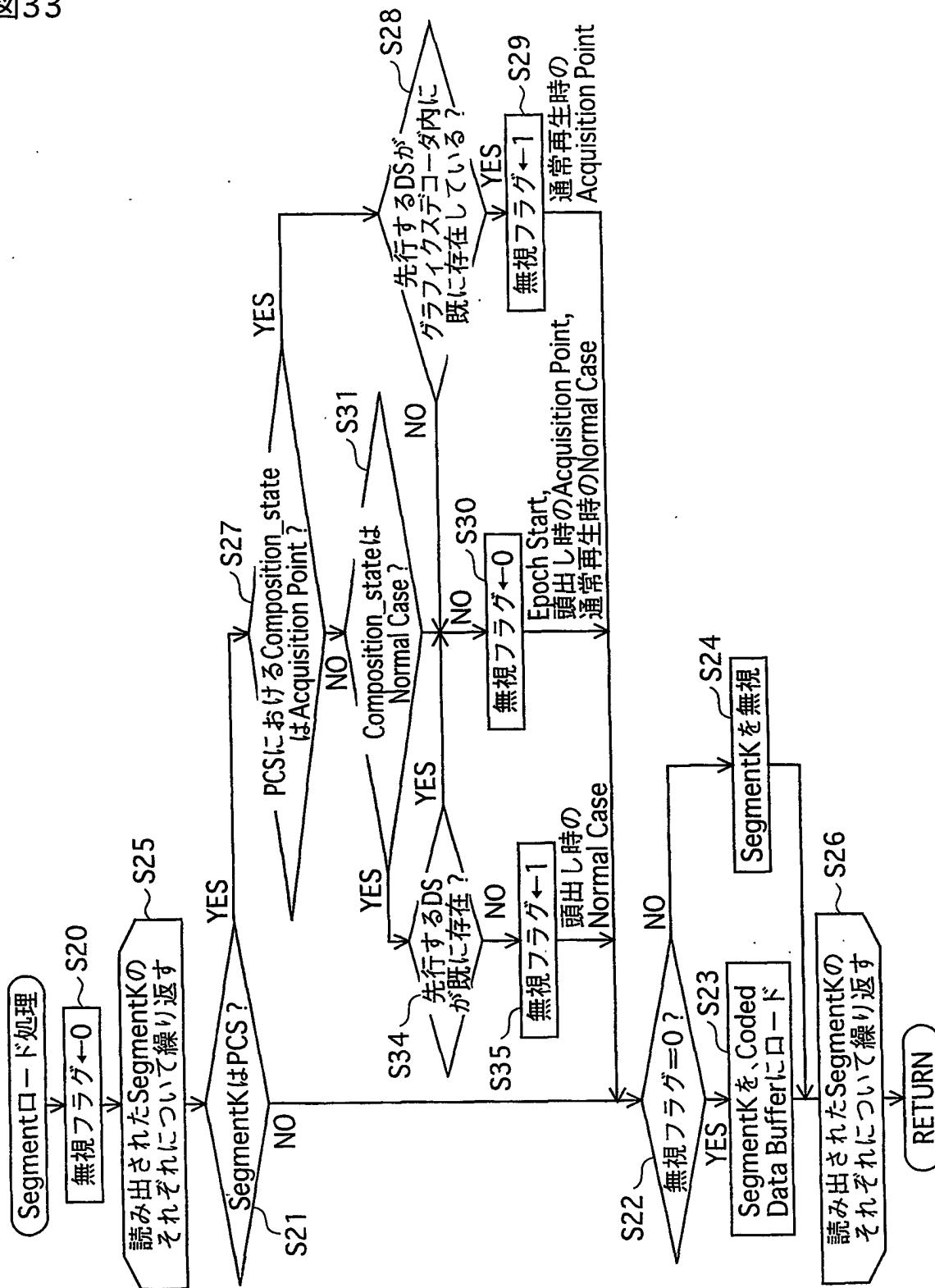


図34

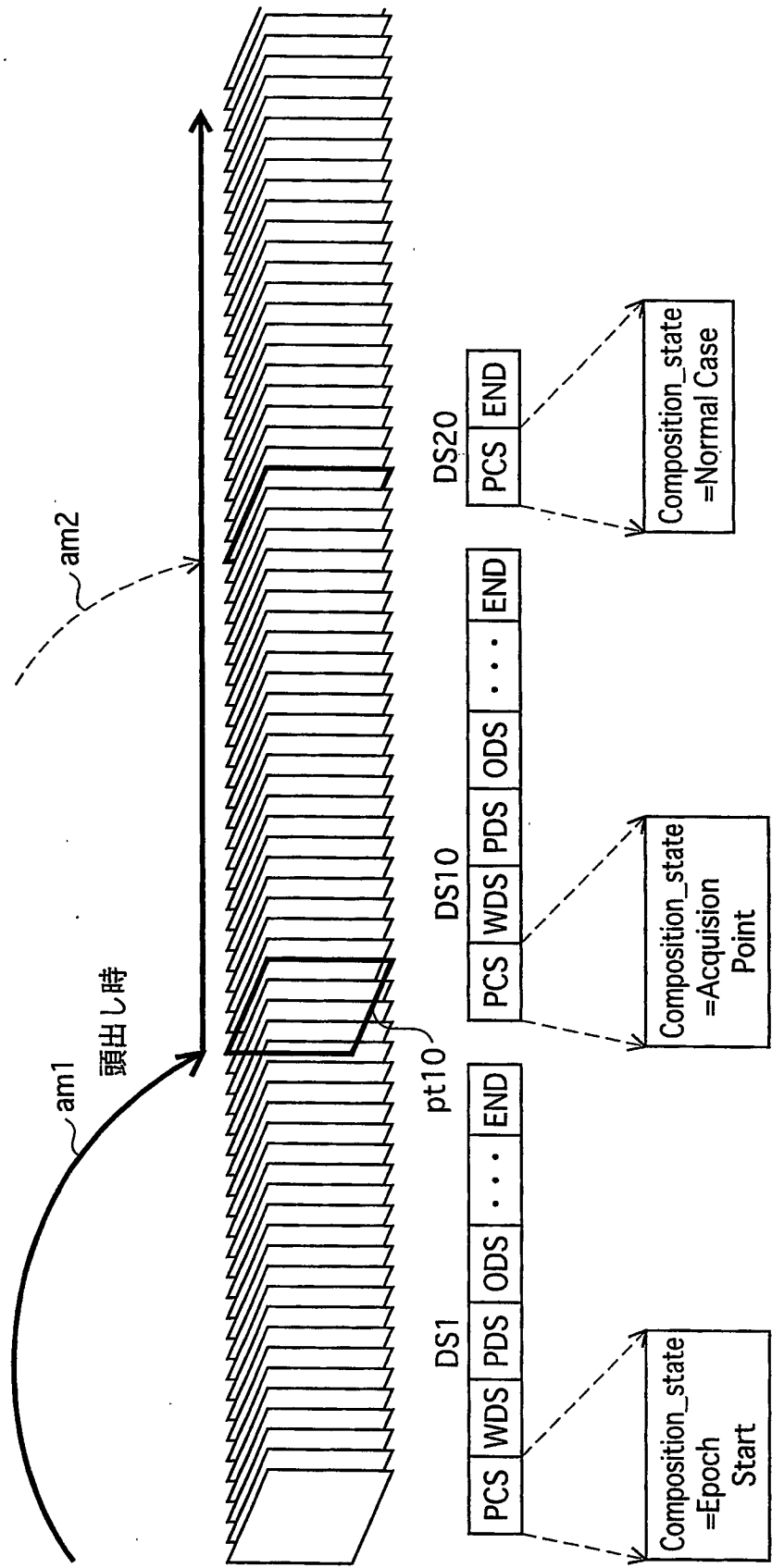




図35

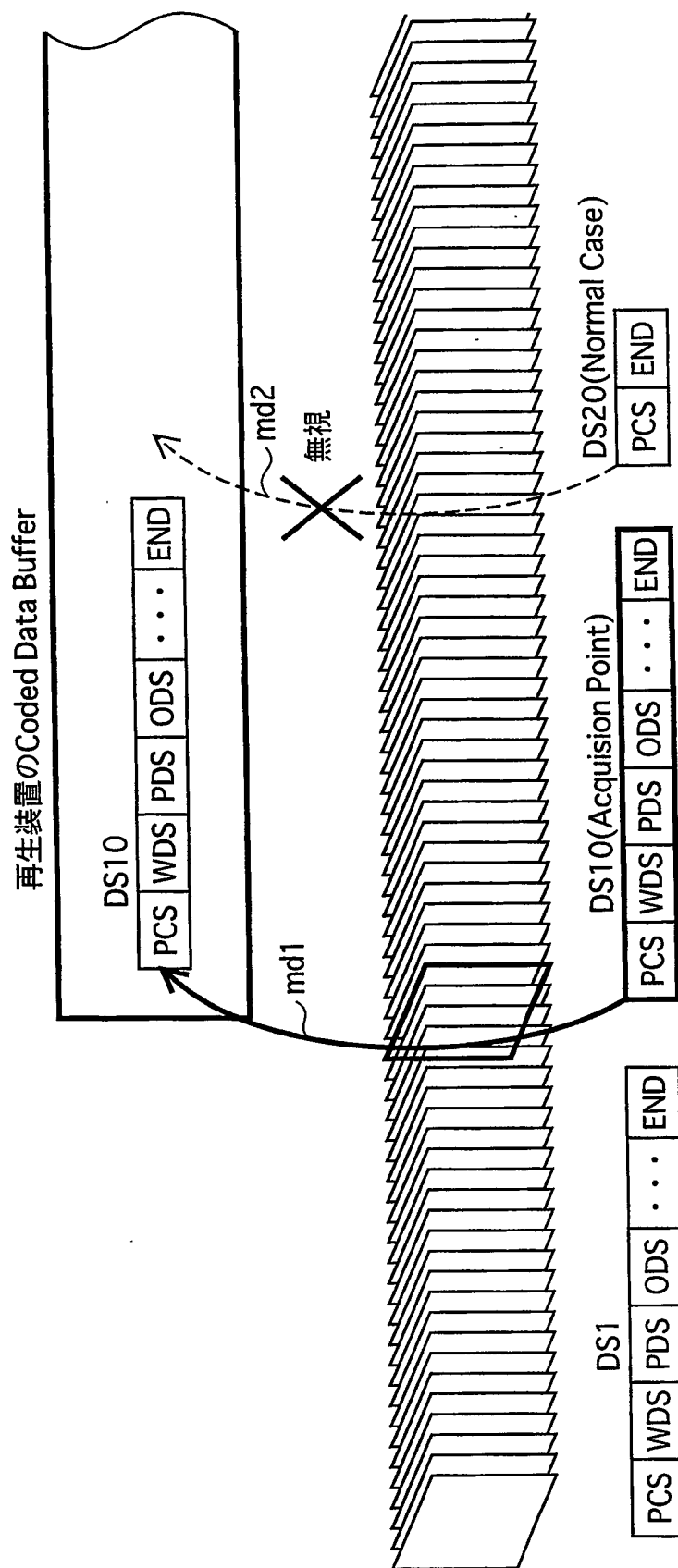


図36

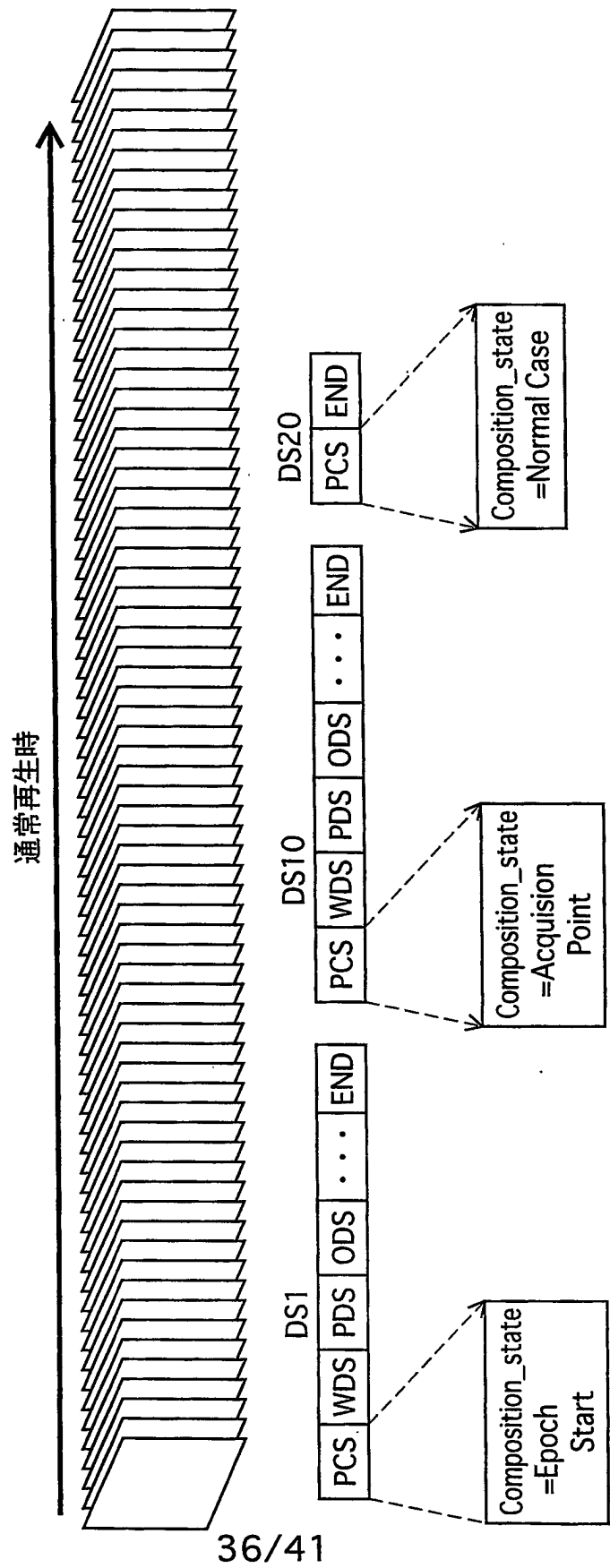


図37

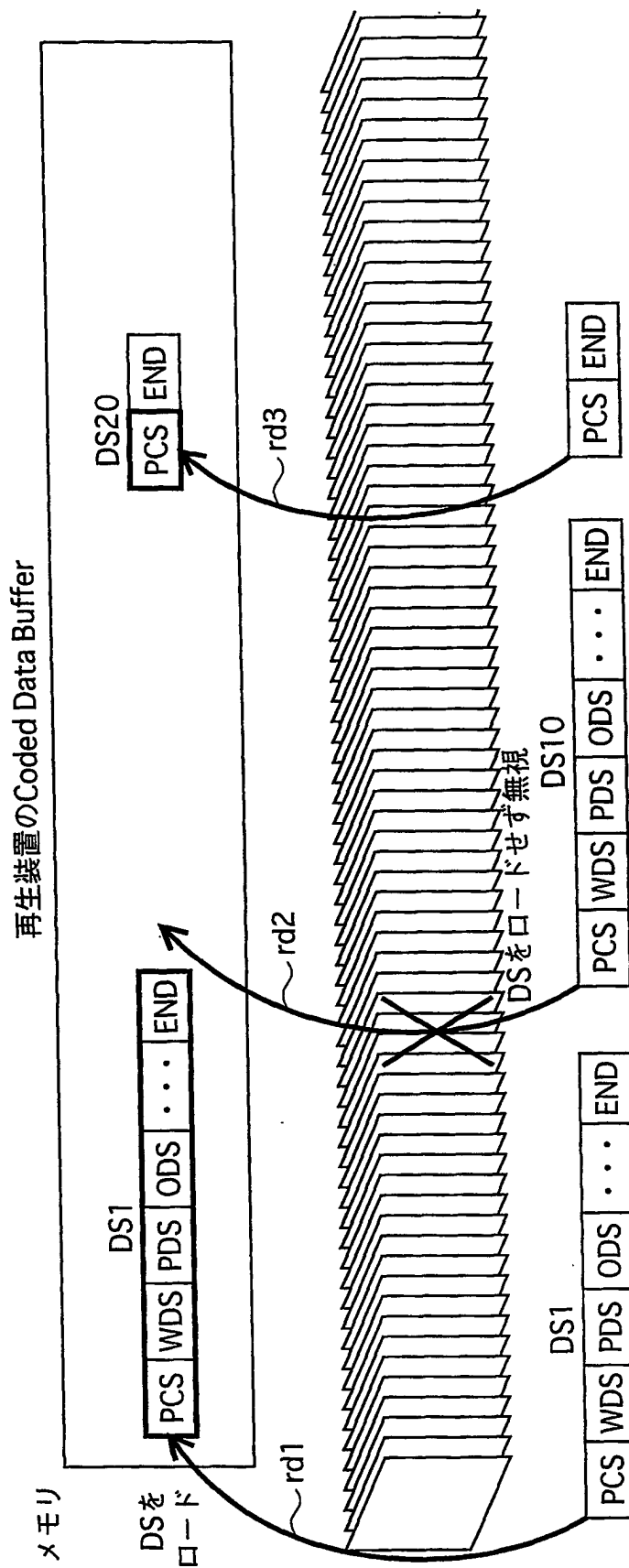


図38

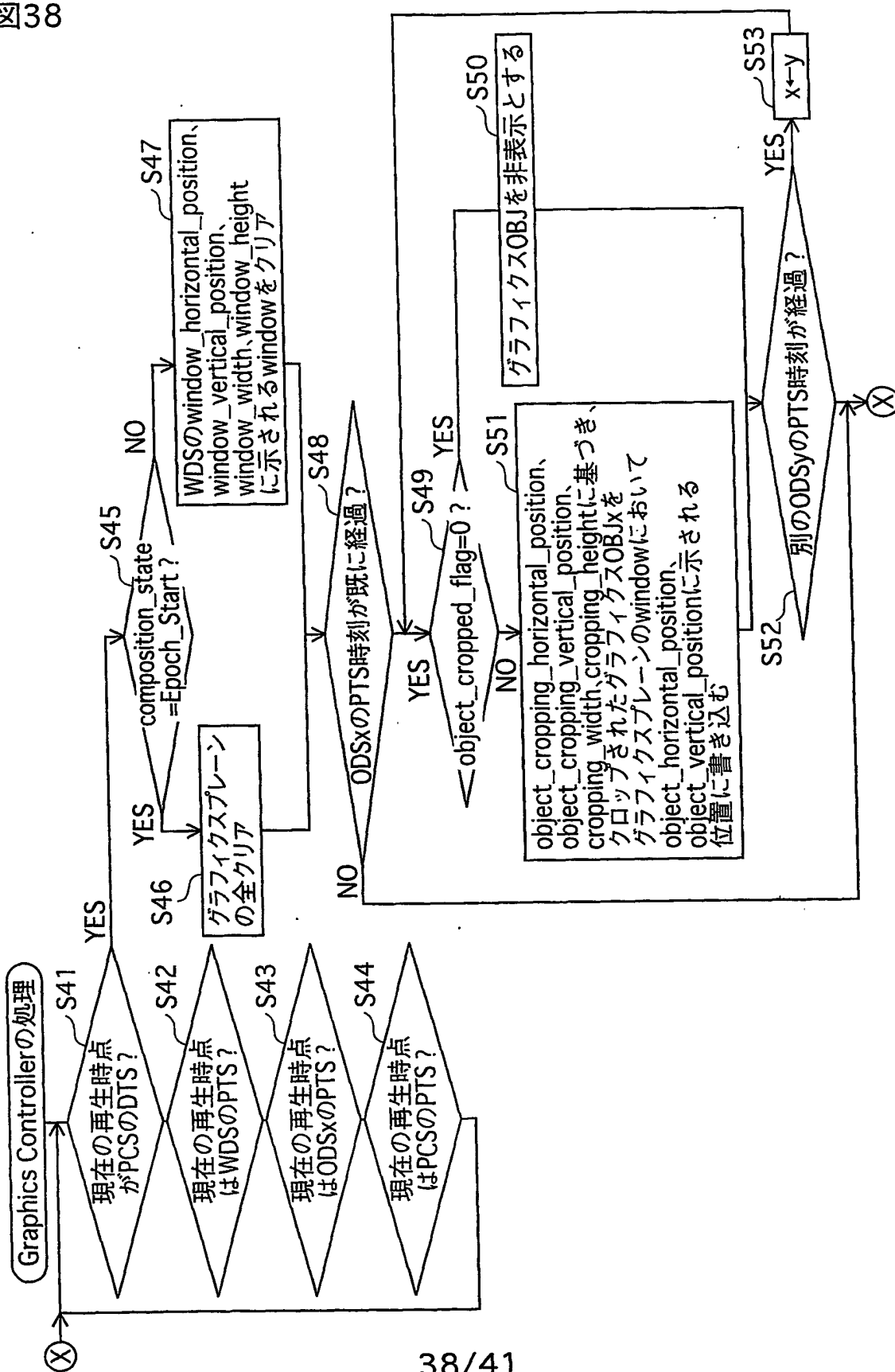


図39

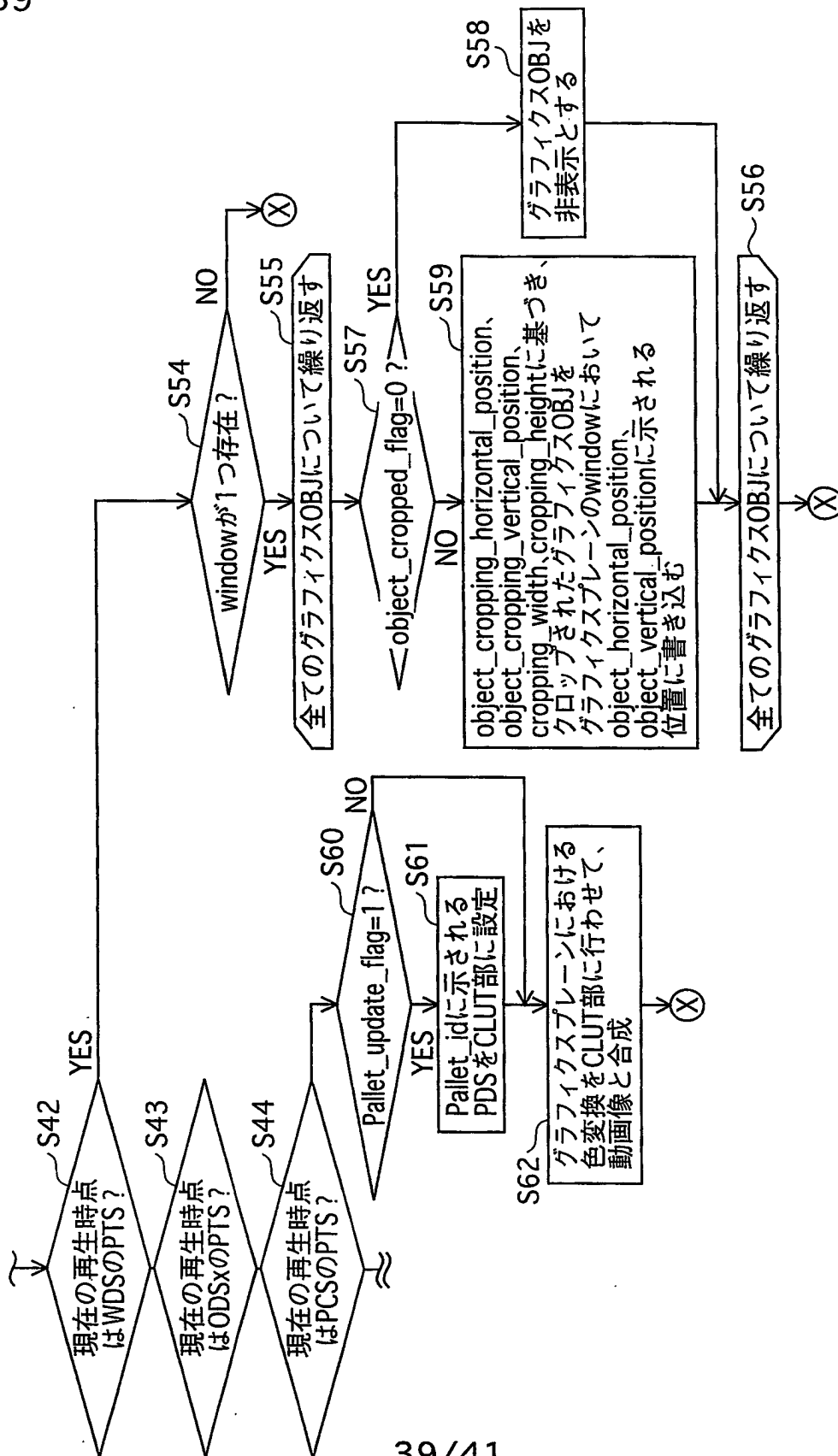


図40

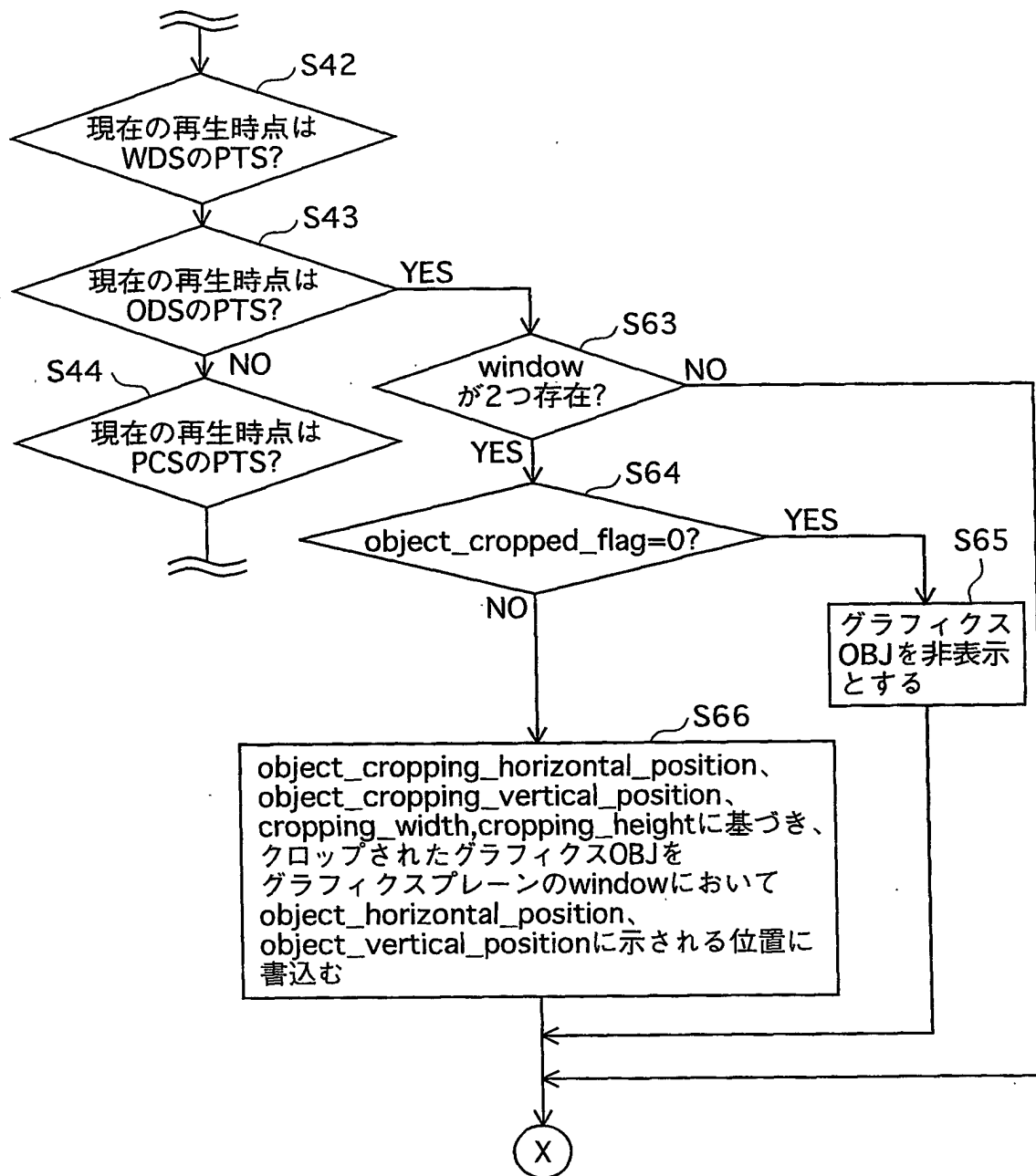


図41

